ST-LOG

THE ATARI ST MONTHLY MAGAZINE

U.S.A. $3.50
CANADA $4.75

ISSUE 10
JANUARY 1987

EXCLUSIVE:
The inner workings of Xanth's super *new* 3-D demo

0 74470 19938 0 01

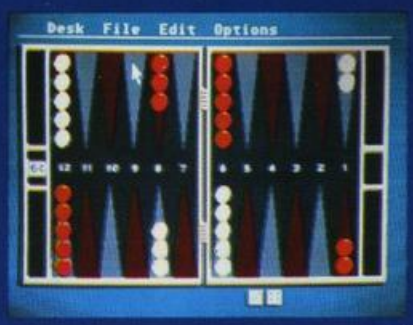
ST·LOG
THE ATARI ST OPERATOR'S MAGAZINE

ISSUE 10
JANUARY 1987

FEATURES

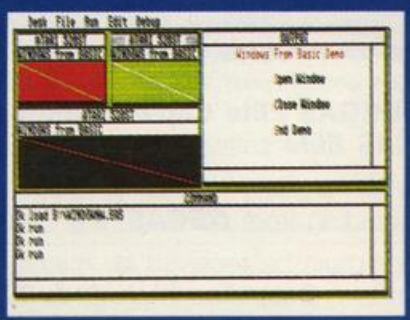
REVIEWS

COLUMNS



9

11



15

Let your artistic nature bring you fame and fortune

in the

D·E·G·A·S Elite

DESIGN & ENTERTAINMENT GRAPHIC ARTS SYSTEM

CLIP-ART CONTEST

Co-sponsored and judged by
ANALOG Computing/ST-Log

FIRST PRIZE
$1000
and your choice of
3 software packages
from
Batteries Included

SECOND PRIZE
A Supra 10-meg hard disk
and your choice of
3 software packages
from **Batteries Included**

THIRD PRIZE
Double-sided drive or
monochrome monitor
and
3 Batteries Included software packages

3 RUNNERS UP
1-year ST disk subscription to ANALOG/ST-Log
and
1 Batteries Included software package

The winning illustrations will be published by Batteries Included in a Clip-Art collection disk, and will be published individually on disk versions of **ST-Log**, *as well as used for illustration in the pages of* **ANALOG Computing/ST-Log** *magazine.*

The **DEGAS Elite Clip-Art Contest** is open only to registered owners of the ***DEGAS Elite*** software program from Batteries Included.

Look for specially marked packages of ***DEGAS Elite*** for your entry card and rules. If upgrading from ***DEGAS***, ask Batteries Included for a **Clip-Art Contest** entry form.

Entries must be received by midnight, EST, May 31, 1987. Direct any inquiries about the contest to: **Batteries Included**, 30 Mural Street, Richmond Hill, Ontario, Canada L4B 1B5

Pick up Elite *and get your mouse in motion!*

READER SERVICE

Use the convenient card on the last page to request FREE information about the products advertised in this issue of **ST-Log**.

For information about specific advertisers or products, circle the number on the card that matches the number shown with the advertisement, or in the Index to Advertisers.

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

The ST-LOG #10 diskette contains 22 magazine files. They are listed below.

FILENAME.EXT	FILE TYPE	COMMENTS
\CMANSHIP\		
TEXTDEMO.C	C	C-manship, listing 1
TEXTDEMO.PRG	RUN FILE	C-manship, compiled
\FONTTRIX\		
FONTTRIX.ACC	ACCESSORY	Font Tricks accessory
FONTTRIX.S	ASSEMBLY	Font Tricks source
OLDE .FNT	FONT	ST font
CHUB .FNB	FONT	Ported 8-bit font
\SLIDERS\		
SLIDERS .PRG	RUN FILE	Sliders demo program
SLIDERS .C	C	Sliders source code
SLIDERIN.C	C	Sliders source code
SLIDERS .H	C	Sliders header file
SLIDERS .RSC	RESOURCE	Sliders resource file
SLIDERS .DFN	DATA	Sliders RCS file
\SPELLER\		
SPELLER .PRG	RUN FILE	Spellbinder program
SPELLER .S	ASSEMBLY	Spellbinder source code
SPELLER .DIC	DATA	Starter dictionary
\WINDOWS\		
WINDOWS .BAS	BASIC	Window BASICs listing
\SHINY\		
SB .PRG	RUN FILE	Shiny Bubbles program
SB .DBA	DATA	Shiny Bubbles data
\DRFLOPPY\		
DRFLOPPY.TOS	RUN FILE	Disk editor
DRFLOPPY.DOC	TEXT	Editor instructions
DRSECT .S	ASSEMBLY	Editor source code
DRTITLE .S	ASSEMBLY	Editor source code
ARCX .TTP	RUN FILE	Unsqueezer program
README .DOC	TEXT	Disk instructions
UNSQUEEZ.DOC	TEXT	Unsqueezing instruc.

Disk instructions:

In order to conserve space, most of the files on the disk have been squeezed and must be unsqueezed before they may be used. We apologize for this small inconvenience, but it was necessary due to the large number of files that had to be included. Please read the UNSQUEEZ.DOC for unsqueezing 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.

WARNING: Be sure to read the appropriate magazine article before attempting to run magazine files. Failure to do so may yield confusing results.

.EXT	DESCRIPTION
.BAS	Requires ST BASIC
.C	Requires C compiler
.S	Requires 68000 assembler
.LOG	Requires Atari ST Logo

ST-Log Staff

Editors/Publishers
Michael J. DesChenes
Lee H. Pappas

Managing Editor
Diane L. Gaw

East Coast Editor
Arthur Leyenberger

Midwest Editor
Matthew J.W. Ratcliff

Contributing Editors
Ian Chadwick
Arnie Katz
Bill Kunkel
Steve Panak
D.F. Scott
Joyce Worley
James D. Yee

Technical Division
Charles Bachand
Clayton Walnum
Douglas Weir

Production
Connie Moore
Edythe Stoddard
Jane Sullivan

Advertising Manager
Michael J. DesChenes

Circulation Manager
Patrick J. Kelley

Accounting
Robin Levitsky

Production/Distribution
Lorell Press, Inc.

Contributors
Charles F. Johnson
Tom Hudson
James Luczak
Steve Rehrauer
Kirk Stover
Rick Teverbaugh

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

Where to write

All editorial material (programs, articles, letters and press releases) should be sent to: Editor, **ST-Log**, P.O. Box 23, Worcester, MA 01603.

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ST-Log**, 100 Pine Street, Holmes, PA 19043, or call 1-800-345-8112 (in Pennsylvania, call 1-800-662-2444).

Correspondence concerning a regular column should be sent to our editorial address, 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.

Advertising Sales

East Coast Gerald F. Sweeney & Associates — (212) 242-3540
P.O. Box 662, New York, NY 10113

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

ST-Log offices Michael DesChenes, National Advertising — (617) 892-9230

Address all advertising materials to: Michael DesChenes — Advertising Production, **ST-Log**, 565 Main Street, Cherry Valley, MA 01611

Permissions

Subscriptions

ST-Log, P.O. Box 625, Holmes, PA 19043; or call toll-free: 1-800-345-8112 (in Pennsylvania, 1-800-662-2444). Payable in U.S. funds only. U.S.: $28.00-1 year; $52.00-2 years; $76.00-3 years. Canada: $36.00-1 year; $68.00-2 years; $99.00-3 years. Foreign: $39.00-1 year; $72.00-2 years; $104.00-3 years. 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.

Editorial

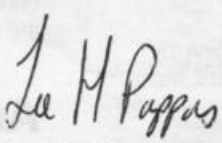
Long ago, in the first issue of **ANALOG Computing**, I wrote with the hope that our publication's name would become synonymous with high quality, and would earn the respect of the Atari community. I feel that we have achieved this goal, and by the mere fact that we're still here—and growing—I think our readers agree. **ANALOG Computing** was the first Atari publication in the marketplace. When Michael Des Chenes and I started work on the premier issue, fewer than 10,000 Atari computers had been sold. We were confident that Atari could build on the fine product they had designed and manufactured. Under the new management, the company continues in that manner today, with the fast-selling ST computer line.

Now **ST-Log** is a monthly, separate magazine. In its pages, we intend to bring you the finest, most up-to-the-minute information to be had. Our goal is to provide ST owners with really useful information, as well as insights that will spark the initiative to utilize the ST for applications above and beyond those which prompted its purchase.

Some of our topics in the near future will include: the widening use of the ST in the business environment, the bright future of desktop publishing, and the increasing importance of the ST in the MIDI/music world. We will be giving those whose ST is their first computer practical help in getting the most from it. And, of course, in our pages you'll find the best software available, for whatever application you require.

The future of the ST looks bright indeed. There is a software surge on the horizon in every field, with major emphasis on CAD/CAM and desktop publishing, as well as all aspects of business needs. There has also been increased talk of CD ROM use, which will provide an added boost to an already exciting market.

The editors of **ST-Log** have attended and participated in many of the Atari users' group shows and professional product shows around the country. We have sponsored our own programming contest for the ST and a joint "clip art" contest with Batteries Included, a leading producer of quality ST software. Staying in touch with the Atari community is vital to our existence. We hope you can see the influence of this interest and interaction in the pages of our latest publication, **ST-Log**.



Lee H. Pappas
Publisher
ST-Log

Reader comment

Notes and footnotes.

I left the worlds of IBM and CP/M when introduced to the ST and subsequently purchased the computer. I was, and still am, thrilled to be in at the beginning of ST computing.

The tutorial articles by Clayton Walnum are an answer to a prayer. At a point in my ST ownership, I was beginning to despair that perhaps already I was too late to "start at the beginning." But the articles on C and interactive fiction have calmed my fears. Inquiries to the developers about the how-tos of computer fiction had resulted in nothing concrete. Then, lo and behold, Clayton Walnum's articles began to appear in **ANALOG Computing**. I was delighted.

And please, continue your editorials. They add such a nice touch, and illustrate the team spirit that results in a single fine product each month.

On another topic: I am sure that most of the editorial staff and the contributing writers to **ANALOG Computing** at some time in their lives had to write papers which needed footnotes. To my knowledge, there is only one word processor that has addressed this need—**Final Word**. Unfortunately, **Final Word** as it appears now is not a satisfactory application program for the ST. Its virtual memory is not really necessary on the ST and causes the drives to go into (what seem to be) spasms of "thrashing"—something I haven't seen since I left 64K CP/M. In addition, on-screen underlining produces letters from the Greek alphabet. Key sequences such as: *[@pagefooting (), @pageheading (right = "value[page]")]* and other such needless nonsense in light of the ST's capabilities is patently absurd. *But it's the only program that footnotes!*

I'm willing to bet that unless the computer is being used by an accountant, its prime use is word processing. Although the trend in scholarly writing is moving toward in-line APA documentation, many students still need to footnote. I have lost (and I take it personally) two colleagues to IBM, MultiMate and Word Perfect, because our university departments require footnoting and there are no satisfactory word processing programs filling this academic need in the Atari program library.

I have faith in Atari. I am sure the company is aware of the problem and is looking into it. I certainly hope so, because I am sitting on my doctoral dissertation right now, waiting for the time I'll be able to enter it into a word processor that will enable me to footnote properly and effectively.

I also have two more colleagues who are about to purchase IBMs because there are a couple of footnoting word processors for that machine. These people are going to spend *thousands* of dollars more than they have to, just because they need the ability to footnote!

Sincerely,
Robert Randall
Kew Gardens, NY

A new breed.

For several years, inspired by my hacker son's enthusiasm, I read every available computer publication from cover to cover, with special emphasis on Atari magazines, because his equipment is Atari. But I was not really interested in owning my own computer. And then the Mac happened. And suddenly there was a computer that sounded as though I might be interested—aimed at people who wanted to use one *for* a variety of activities, rather than *as* an activity. After almost a year of meticulous and painstaking comparisons, I bought an Atari 1040ST.

I was (and still am, to an extent) like a teenager with his first car. And when I read that several magazines were about to bring ST-oriented publications on the market, I was quite enthusiastic, and planned to subscibe to the lot of them.

Then I saw *STart* and read the ST inserts in **ANALOG Computing** and *Antic*, and I began to think that perhaps neither the editors of any of the three magazines (nor even Mr. Tramiel himself) understood why I had bought my ST. I got the same reaction from several other ST owners. I discovered, to my surprise, that the SYSOP of what he hopes will be the focal ST BBS in the Southeast did not bother to buy a copy of *STart*.

Why? We are enthusiastic about our computers, and we are anxious to learn how to get the most from them.

Several days ago, desirous of using my new QMI modem, I tried logging on to their BBS. I was greeted at the outset with what to me was an entirely inscrutable question about NULLS. No button I pushed got me beyond that command, so I quit. The modem instruction book doesn't mention NULLS. The terminal program instructions don't mention NULLS, and the 1040 instruction book doesn't mention them, either. QMI seems to assume a level of hackerdom that I don't have. . . A level that I do not want!

And there, it seems, is the gist of the problem. I bought an ST because I am a sometime writer and wanted the word-processing facilities. I bought an ST because I am a sometime artist and wanted

ATARI ST

PUBLISHING PARTNER™

CREATES LIKE A PUBLISHING COMPANY WITHOUT THE OVERHEAD

SOFT LOGIK NEWS

PUBLISHING PARTNER ™ HELPS YOU CREATE!

AVAILABLE FOR ONLY $149.95

You'll benefit by using your Atari ST to create professional quality journals, newsletters, ads, business cards, certificates, letterheads, logos,art designs, bar graphs, flow charts, *even bumper stickers* and all the forms you or anyone would ever need. Create just like a professional publishing company without the overhead! Publishing Partner is actually three progams in one--Word Processor, Page Layout, and Forms Creator. Expand your potential in becoming a better writer, artist and designer with your Publishing Partner.

WORD PROCESSING

What you see is what you get!
Combine text and graphics easily and quickly from existing or newly created documents. Position entire paragraphs or individual words exactly where you want them. Create one, or multiple color separations ready for printing.
Production time will never be the same--it will be much shorter!!
Just take a look at some of its features!

- GEM based
- Justifies right or left as you type
- Edit Multiple Columns on One Screen
- Search and Replace
- User Definable Page Size
- **Bold**, underline, ^{super} and $_{subscript}$
- *Italicize*, **shadow, outline**
- **Reverse Image**
- Backslant Characters
- Mirror Image • Invert Image
- Auto Headers/Footers, Page Numbers
- Easily Move Text
- Reads & Merges other files together
- Sets Tabs
- Macros
- Vertical & Horizontal Printing

PAGE LAYOUT

Whatever you require--cutting artwork from other programs, custom logos, unique borders, unusual mastheads, digitized photos--Publishing Partner is your solution. After all, it was specially designed for you--the home and/or office Atari ST user--by the pro's who realize that there's more to your computer than just typing letters.

- Vert. and Horz. rules--Exact Alignment
- Auto Text flow for columns/pages
- Layout Multiple Columns of text
- Change columns on finished page
- Display entire page, 50%, or 25%
- Easily Re-position Text and Graphics
- Alternating Headers/Footers
- Mix Type, Fonts, and Sizes *anywhere*
- Adjust Sizes from 2pts to 144 pts (2")
- Adjust line spacing (leading) by points
- Import other program's graphics
- Rotate Graphic Images
- Multiple Patterns, Shades and Colors
- Enlarge & Reduce Graphics/Exact fit
- Cut, Paste & Crop Graphics
- Tool box function/Unlimited patterns
- Boxes, Circles, Arcs, Polygons, etc.
- Insert lines directly on the page
- Adjust Character Spacing
- Use Hairlines to separate columns

FORMS CREATOR

You can create a variety of forms quickly and easily with your Publishing Partner. For example, you can create your own:

Letterhead
Invoices
Purchase Orders
Labels
Bumper Stickers
Business Cards
Certificates
General Ledger
Shipping and Receiving
Routing Slips
"While you were out" Phone messages
Templates
Price Estimate forms
Requistion forms
Shipping Logs and much more!

Publishing Partner supports most dot matrix printers, including the Epson ™ Star ™ and Okidata ™ Printers. Also supported are any Postscript output devices such as the Apple Laserwriter ™ New print drivers are constantly being released, *so please call to make sure your printer is supported.*

FOR MORE INFORMATION OR TO PLACE AN ORDER, CALL (314)894-8608.
DEALER INQUIRES INVITED.

Soft Logik Corp.™

4129 OLD BAUMGARTNER * ST. LOUIS, MO. 63129 * CALL (314) 894-8608

CIRCLE #103 ON READER SERVICE CARD

its graphics and color capabilities. I bought the ST because I am a part-time musician and want its musical potential. I want to use my computer to enhance my jobs, my hobbies and my life.

I do *not* want to be a programmer. I do *not* want to be a hacker. I don't care about C! I don't wish to know anything about Logo. I am interested in programs I can use, not write. I am interested in reviews of activities and programs that will help me to use my computer as a tool, not as a hobby in itself. I am not at all interested in games—I can find more amusing things to do than shoot down spacemen or play solitaire.

We ST-Amiga-Mac owners seem to be a breed of user the old guard has not yet understood. We are not business men, seeking ever grander databases (I am only mildly ashamed to admit: I really don't know what a database is—but, after several years of intense reading of all the leading periodicals, I have never gotten a hint from one of them); we are not tinkerers, dying to try out yet another word processor; and we are definitely not hackers.

A recent editorial in one of your magazines deplored the lack of software sales and hinted strongly that ST owners were engaged in pirating like mad. I suspect the truth is the opposite. I am willing, and can afford, to buy software. But at the prices asked, I want to know something about it before I buy. I hope for reviews, but get yet another compiler program; I hope for ads that tell me something about the product, but get minimal information, couched in the inexplicable language of the experienced hacker. I hope for materials to help me expand my computer use, but get pages of games and programs to help me copy copy-protected disks. I learned by chance that **1st Word** has been improved since I got it—but Atari didn't tell me.

I realize that part of the problem is a kind of circularity built in to the current computer magazines. The magazines offer many programs to their readers. And, in order to have programs to print, they must encourage readers to write these. But I have a feeling that we ST-Amiga-Mac types are not going to be program writers—although we will be quite interested in programs. . . if they're on disk.

The ST is not an 800XL. It is not an IBM. It is not a neighborhood arcade. I look forward to a new generation of editors and manufacturers who see computers as ST-Amiga-Mac people see them—useful enhancers of life, not business tools or hobbies.

Sincerely,
Robert J. Fusillo

*Funny you should mention it. . . in this, the first fully separate **ST-Log** issue, we're "catching up" on reviews. And we hope to have regular "how-to" features for ST first-computer owners in the near future. As **ANALOG Computing** is for all Atari lines, so **ST-Log** is for all ST owners.* —*Ed.*

Classical Gaming

The great strategy games come to the ST.

by Bill Kunkel, Arnie Katz and Joyce Worley

Computers have been around since the 1940s, when they were developed to crunch big numbers and break ciphers. The first experience with bits and bytes for most Americans, however, came when they played tic-tac-toe against a micro.

Competing against a machine, especially in a game which previously required two players, never fails to impress. And as memory limitations for home computers eased, it was only natural that other non-electronic strategy games would migrate to the new medium.

Chess, checkers, and the like have stood the test of centuries. Even people who play no other games can be coaxed into a quick chess match or a few hands of poker. These games have proven doubly popular on the computer for several reasons: (1) classic games are an oasis of familiarity in the often unfamiliar world of microcomputers; (2) the computer eliminates the need to find an adversary of equal skill; and (3) for all-skill strategy contests like chess, reversi and checkers, the computer is a faster, less complicated alternative to postal play.

So it's not too surprising that the classic strategy games are a staple of every system's software library. In fact, they're often included in the first wave of titles which appear for a new brand of computer.

Admittedly, this has as much to do with the logistics of the software business as the inherent value of the games. There's always a gap of months between the debut of a new computer system and the arrival of a large quantity of software.

For users, this "waiting window" is a time of frustration. To ease this pain, publishers generally try to get something into the stores quickly, while they start original designs down the development pipeline.

The two most frequent choices are text adventures and classic strategy games. Both types of games have established followings and are easy to port over to new systems. Since owners of the Atari ST, like users of other 68000-based machines, have shown little inclination to buy text adventures, the classic strategy contests have come to the fore even more strongly than when the 6502-based systems first hit the market.

Chess is the game which first comes to mind when discussing classic strategy contests that adapt well to computer play. Amazingly enough, no commercially produced chess programs are available for the ST at this writing. (Public domain chess programs are obtainable.) Obviously, publishers are reluctant to produce a chess game unworthy of this new, high-powered computer medium.

Worry not, pawn-pushers! The folks of Software Country are about to fill the "chess gap" in spectacular fashion, with **Chessmaster 2000** from Electronic Arts, 2755 Campus Drive, San Mateo, CA 94403 — (415) 571-7171. Not yet available, this state-of-the-art chess program will accept pawn promotions, allows players to castle, recognizes pawn captures en passant, and contains an extensive library of openings. **Chessmaster 2000** acts as referee during two-player encounters, plays demonstration matches, or competes against a human player.

The most impressive thing about **Chessmaster 2000** is contained in its name: its manufacturer claims that the *2000* refers to its player rating, according to the International Chess Federation (FIDE) system. This puts it at the Expert level, and only 300-400 points below the ratings held by most International Masters (the top Grandmasters usually have ratings of 2500-2650).

The "chessist" can compete at any of twelve different skill levels, change sides at any time, and determine how many moves the computer can plan ahead.

All of these features are nice, but what should turn heads is the program's gorgeous graphics. Each piece is beautifully sculpted, with tones and colors so realistic they look as if they could be picked up and moved by hand.

Publishers often sell a chess program by touting the number of Grandmasters and other programs it has outplayed. "It whipped **Sargon**!" is a commonly heard boast, but since only a minuscule percentage of users can compete at that level, this is a dubious selling point. Ease of play, good graphics and the options menu are more important, and **Chessmaster 2000** is strong in all three areas. Watch this space for more details.

Backgammon seems more fashionable than chess these days. The ST boasts what is probably the finest program of this type ever produced for a home computer, **Hippo Backgammon**. It's from Hippopotamus Software, 985 University Ave., Suite 12, Los Gatos, CA 95030 — (408) 395-3190, priced at $39.95.

The world of gammons, primes, blots and doubling cubes makes a slick translation to the computer. The programmers at

Hippopotamus even went beyond the call of duty and included an excellent primer on Artificial Intelligence in the bargain. For a complete review of this product, see **ST-Log** issue 5 (in **ANALOG Computing**'s issue 45).

Backgammon fans looking for something a little more modest in price should check out **Peggammon**. This program displays the game board in overhead perspective, and the "points" are depicted as spools holding the donutlike pieces. There are no pull-down menus, but a command window is brought up by clicking the right mouse button.

Options include one or two-player modes, a pair of difficulty levels (experienced and beginner) and an "Edit" command that permits users to realign the board at any time. The graphics are pleasant, featuring green and silver-colored points, but the various tables and the bar are not rendered in any detail. Nonetheless, it is a reasonably priced, solid representation of this classic contest.

Peggammon is from Artworx Software Co., Inc., 1844 Penfield Road, Penfield, NY 14526 — (716) 385-6120, $17.95.

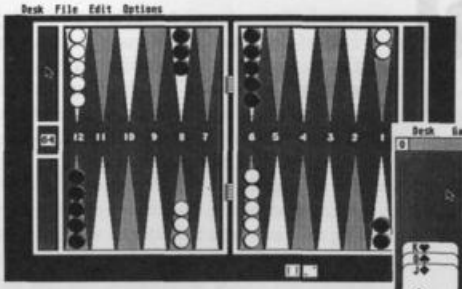
Reversi is another game that's been around a long time. The ancient Chinese played it in its original incarnation, Go. More recently, the game was repackaged as Othello and has again become a staple of the board game universe.

Reversi is, at once, both elegantly simple and diabolically complex; it can be learned in minutes, yet mastery may take a lifetime. It also adapts perfectly to computer, as proven by **Flip Side**, by Ken Olson and Phil Hollyer, from MichTron (576 S. Telegraph, Pontiac, MI 48053 — (313) 334-5700, $19.95). **Flip Side** uses the GEM interface and ST graphics to produce a game that is as compelling today as it must have been to the Orientals of antiquity. One player uses dark disks, the other, light. A player can "flip" an entire column of enemy disks to his color by trapping it between a pair of his own. Sounds simple? Heh heh heh.

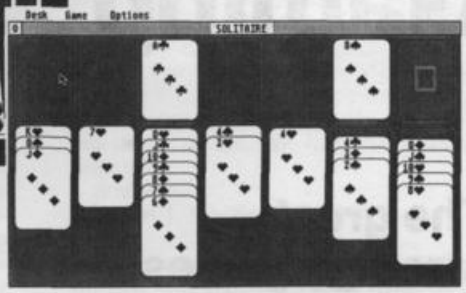
The mouse interface handles everything. The player clicks on the desired square on the gridlike board, and his piece appears, causing the captured disks to change, domino fashion, to their new color.

The pull-down menus let players change sides, inspect all available moves and even edit the board. Heck, the computer's such a good sport it'll even suggest a move if you're stuck (an excellent way to learn the game, by the way). **Flip Side** offers two-player and solitaire versions, with six levels of difficulty.

Card games such as poker and blackjack may not possess the exalted bloodlines of the previously discussed strategic amusements, but they're nonetheless quite popular. More beloved still, however, is that variant of poker in which the objects one gambles with possess a more intimate value than mere money.



Hippobackgammon.



Electro Solitaire & 21.

Although it has been produced for just about every computer system short of the one that runs NORAD, **Strip Poker** is the best it's ever been in this ST translation by Arthur Walsh and Todd Kepus. Published by Artworx Software Co., Inc., 1844 Penfield Road, Penfield, NY 14526 — (716) 385-6120, $39.95. The reason it's better is twofold: the mouse makes for a slicker and speedier interface device than a joystick, and, even more significantly, the Doug McFarland graphics look nicer than ever!

The game disk includes both "Suzi" and "Melissa," an incorrigible flirt in a cutoff top, and Artworx promises forthcoming data disks with opponents of both sexes. The girls still say funny things ("What have you been smoking?" after a dumb play; or "Be still my heart!" when the user is forced to disrobe), as in earlier versions, and play a pretty decent game of poker. The program is easier to bluff than it should be, but then, no one is looking for a tough contest here. The object is to get the ladies out of their clothes, not to lose one's own, after all.

The playing system is simplicity itself. The player clicks once to ante ($5.00), then either bets, stays or drops. If both sides stay, the hand is a wash and they ante again. Once a hand is played, the user can bet, call or drop at any time, up to three raises. When players lose their available money, they must hock their clothing for a new stake.

Strip Poker is one of those programs with a nice blend of play value and novelty. More conventional gamers, however, may prefer **Electro Solitaire & 21** from Soft Logik (4129 Old Baumgartner, St. Louis, MO 63129—(314) 894-8608, $19.95).

It's a low-priced blackjack and solitaire program that plays a good, quick game of either with a minimum of visual frills. When playing Blackjack, the cards appear on the left side of the screen, with click-on options (hit, hold and, when appropriate, split) on the right. In Solitaire, cards are "dealt," using a simple but effective animation. The player can click on the desired card and drag it to its new location. Both games offer access to option menus.

The "waiting window" has just about closed for the ST. New releases are pouring in which, generally, represent the cutting edge in computer entertainment. However, there are still a few classic contests yet to be exploited by software publishers. How about some Chinese checkers, folks? Or perhaps a Parcheesi program? Heck, there haven't been more than a handful of computerized word games along the lines of Scrabble.

But then, that's the beauty of the game universe: no matter how far you go into the future, some of the best things remain the oldest and simplest.

Reality is a convenient measure of complexity, but why be restricted to reality?

Alvy Ray Smith, Lucasfilm

Shiny Bubbles

Ray tracing by the Xanth F/X division

(or: better demos through modern mathematics).

by James D. Yee

Our newest creation is a twenty-frame animation sequence called **Shiny Bubbles**. In it, four mirrored spheres roll on an infinite field of *ATARI*s under a checkerboard sky. The process used to generate the images is called "ray tracing." This produces a view into a computer simulated universe, presumably the Xanth xone.

The inspiration for this demo came from a computer-generated printout I brought back from COMDEX earlier this year. The printout was from an $8000.00 color thermal transfer printer. (I suspect the computer attached to it was significantly more expensive). In any event, Park got the ray-tracing bug. The first result was a program called Tracey. Tracey generated some **DEGAS** medium-resolution pictures that we used for our store's T-shirts.

What does Tracey do? First, it generates two planes stretching to infinity, one below with the repeated word *ATARI*, and one above with a red and yellow checkerboard. Second, it creates the spheres, which reflect both the planes and the other spheres.

"How was this done?" you ask. Remember, in our simulated world there are only two things you ever see: ground and sky. And no, I didn't forget the spheres; you can't see perfect mirrors (just what they reflect).The ATARI horizon is a **NEO-Chrome** picture laid out in rectangles very much like a checkerboard (actually, the sky *is* a checkerboard). If we were viewing the checkerboard/ATARI field at 90 degrees, we would see the ATARIs with no distortion (*verrrry* boring). At 0 degrees, we would see a horizontal line (even more boring), so we tilted our universe back 45 degrees (for our calculations, 0.79 radians). Tracey distorts the ATARI and reduces size to simulate the appropriate distance. So far, so good; now we have ATARIs *ad infinitum* (see Figure 1).

So. . .we just need to design a little algorithm, based on the formula for the sphere, and to deflect our view to an appropriate place in our little world. Confused? You won't be, after this month's episode of *Ray Tracing!*

Imagine that one ray of light from the scene we're viewing strikes the screen at the exact location of a screen pixel. Each pixel on the screen is struck by one ray of light, and one ray only. The state of each pixel (whether it's on or off, or what color it is) is determined by "its" ray. Now reverse the process: follow each ray of light from its end-

Figure 1. — The planes.

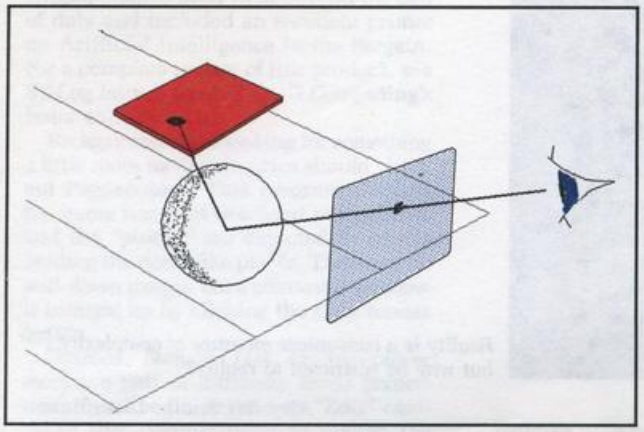


Figure 2. — The rays.

point on the screen back to its source—*not* to the ultimate source of the light, of course, but to the object (here, either the ground or the sky) from which it was "originally" reflected. Tracing each ray in this manner allows us to determine what value its destination pixel should have. In the case of **Shiny Bubbles**, the ray may bounce several times between its destination pixel and the object from which it was first reflected.

Aside from the graphic end of **Shiny Bubbles**, there were a few things that had to be done before we felt it could be released. The first public showing of **SB** was at the San Jose Atari Show. That version used 1 meg of memory and required 1 meg of disk space—meaning it wouldn't run on a stock 520ST.

The San Jose show was fun and gave us a chance to network with a few experienced ST artists. A few weeks later, our home-brew routine compressed the **SB** data to a single 360K disk. Last night, we trained **Shiny** to decompress the animation data and display it on the screen simultaneously.

Picture compression routines typically analyze the screen data and pack the color data as it's stored. For example, if there are large blocks of the same color across the screen, a squeeze routine will count the number of duplicate pixels and then replace their data with a pixel count, followed by the color. Our routine just compares the new picture with the old and stores the changes. So, instead of drawing the entire picture over with a new frame, we just modify the existing picture. This system works well on data that's really similar, and speed is essential.

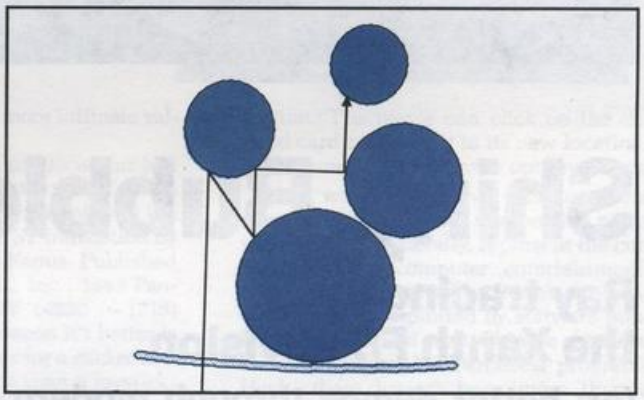
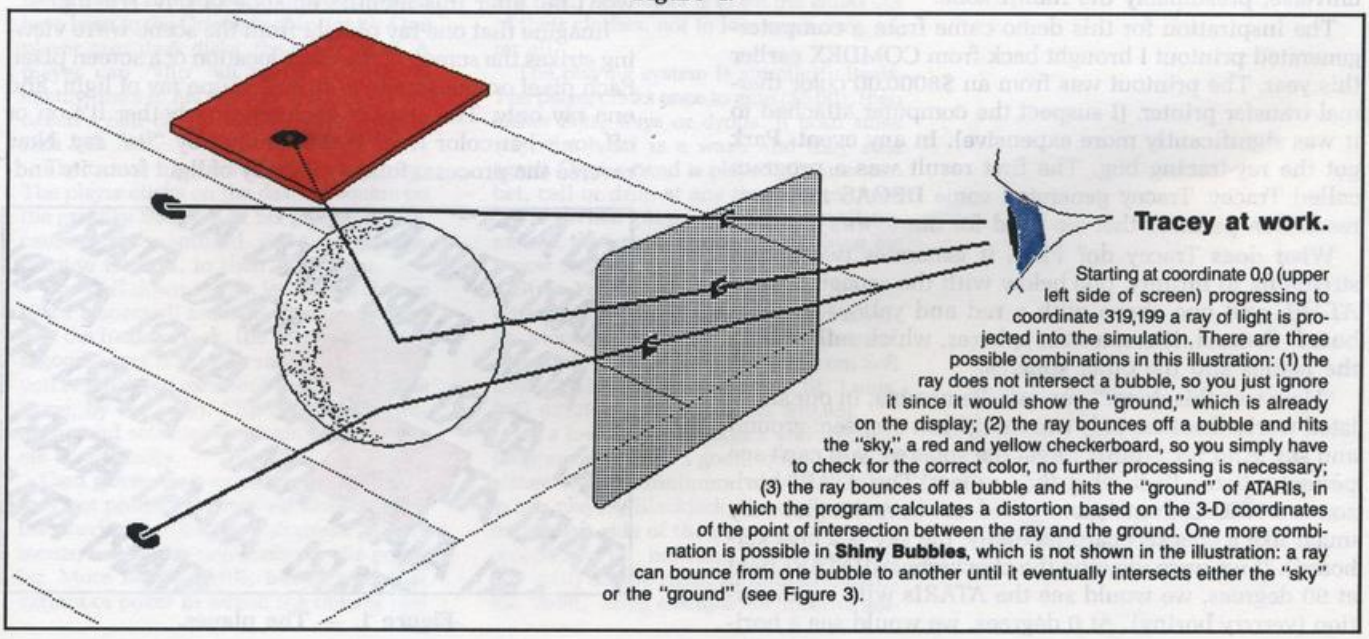


Figure 3. — Top view.

Park tried a couple of other tricks, but they didn't quite live up to our expectations. Anti-aliasing is a technique that smoothes out the jagged edges appearing on diagonal lines (they look like stair steps). The usual method is to reduce the contrast on the edges of the lines. The routine worked, but the screen looked blurry. The name would

Figure 4.



Tracey at work.

Starting at coordinate 0,0 (upper left side of screen) progressing to coordinate 319,199 a ray of light is projected into the simulation. There are three possible combinations in this illustration: (1) the ray does not intersect a bubble, so you just ignore it since it would show the "ground," which is already on the display; (2) the ray bounces off a bubble and hits the "sky," a red and yellow checkerboard, so you simply have to check for the correct color, no further processing is necessary; (3) the ray bounces off a bubble and hits the "ground" of ATARIs, in which the program calculates a distortion based on the 3-D coordinates of the point of intersection between the ray and the ground. One more combination is possible in **Shiny Bubbles**, which is not shown in the illustration: a ray can bounce from one bubble to another until it eventually intersects either the "sky" or the "ground" (see Figure 3).

have to have changed to **Fuzzy Bubbles**, so we dropped that project into the bit bucket.

One technique we kept in the program is luminance loss. It isn't obvious, but **SB** uses twelve colors in the main picture. Images in the foreground are brighter than those in the mirrors, to simulate imperfect mirrors. That didn't leave any colors for Fuji-boink fades, which were in the original design. Oh well, that's life.

What's next from Xanth F/X? Well, it's getting harder to top our previous efforts, but our little multi-player, multi-computer maze game seems to be next on the list. We ran it at the Portland Atari Show on thirteen machines (we ran out of MIDI cables). It was a blast! "Maze" (or "Kill a happy face") is still officially untitled. The object is simple: you're stuck in a 3-D maze with up to fifteen other "smiley faces," and the first person to bust ten faces wins the game.

One of the features we've considered for the Maze game is making it function with the Tektronix stereo glasses, for real on-screen 3-D. We might even break convention and actually make it a commercial program, by having it use a modem instead of—or in addition to—the MIDI port.

Another project we would like to do is to take control of a wall of Atari monitors at one of the trade shows and convert these into a giant video billboard. Can you imagine a four-monitor high and wide spinning Fuji symbol?

We would like to thank C.o.a.s.t., DACE and SLCC user groups for inviting us to the San Jose show, PACE and Capt. Bananna for their hospitality at the Portland show, and all the folks who have patiently waited for copies. As this article hops on the Fed-Ex truck to **ST-Log**, copies of **Shiny Bubbles** will be on the way to their respective new homes (Leonard, you'll get the first copy). ///

There is no person named Xanth Park. Xanth is not a person but a place, and a group of friends. Park is the Xanth programming core. Park is not to be confused with Xerox PARC (Palo Alto Research Center), but has similar research and academic goals. One might say they have the same "look and feel," shades of GEM. Can you say "Apple lawsuit?" Sure, I knew you could. . .

Xanth F/X is a public relations and promotion group that's more like a specialized user group, primarily populated by staff and friends of Xanth Computer Systems Inc. Xanth Inc. is a busy, understaffed Atari-ST-only retail store in Seattle, Washington, that just cloned itself in Bellevue. Between the two stores, they plan to take over the Atari world. Xanth welcomes visitors (especially customers) and letters. Any ideas or suggestions may be sent to us, Attention: F/X division, Park, Sysop X, or Jim (yours truly)—Xanth Seattle, 600 First Avenue, Seattle, WA 98104, or Xanth Bellevue, 14100 NE 20th, Bellevue, WA 98007. The Xanth SST BBS number is (206) 682-8039. The BBS has been in stealth mode lately and needs some callers. I hope you've enjoyed reading this as much as I've enjoyed writing it. SYSOP X.

ATARI ST SPECTACULAR

ATARI 520 SYSTEM PACKAGE*

*Including RGB or Monochrome Monitor, Mouse, Disk Drive, Basic, Logo, Neochrome, 1st Word, TOS on ROM, and RF Modulator

CALL
MONOCHROME SYSTEM

CALL
RGB/COLOR SYSTEM

ATARI 1040 ST SYSTEM PACKAGE*

*Including RGB or Monochrome Monitor, Mouse, Double-sided Disk Drive, Basic, Logo, Neochrome, 1st Word, TOS on ROM, and Built-In Power Supply

CALL FOR CURRENT PRICE

***All ST System Packages are sold with a ComputAbility 90 day warranty.**

SUPRA 20 MEG 3.5 INCH HARD DRIVE
only
$659

ATARI SF 314 DISK DRIVE
Double sided/ 1 Megabyte storage
$199

PRINTERS

NX-10.209	SD-15.419	Powertype.....229
NL-10.Call	SR-10.459	Panasonic 1080i .219
NX-15.329	SR-15.559	Panasonic 1091i..279
SD-10.309		Citizen 120D ....Call

OKIMATE 20 COLOR PRINTER & ATARI ST PLUG & PRINT
$199
This is a shipped price anywhere in continental USA

ST MODEM PACKAGE
NEW AVATEX 1200HC MODEM
1200 Baud-Hayes Compatible
+
ST MODEM CABLE
+
ST TALK TELECOMMUNICATIONS PACKAGE
$159

WE RECEIVE ST PRODUCTS ON A WEEKLY BASIS. IF YOU DON'T SEE IT HERE . . . CALL

SUPER SPECIALS*

Defender of the Crown . 31.95

MINDSCAPE

S.D.I. 31.95

*SUPER SPECIAL PRICES EFFECTIVE NOW THRU JANUARY 31, 1987

ATARI 520 ST • ATARI 1040 ST SOFTWARE

ABACUS
Textpro 37.95
Datatrieve 37.95
Forth/MT 37.95
Paintpro 37.95
Textdesigner 37.95
Assempro 37.95
PC Board Designer 299.95
Abacus Books ..... Call

ST GRAPHICS
Degas 27.95
Degas Elite 52.95
Easy Draw 54.95
PC Board Designer 299.95
Paintworks 49.95
Super Graphics 33.95
New Tech Color Book 16.95
Graphic Artist 149.95
Real Cadd ..... CALL
Aegis Animator ST 49.95

INFOCOM ST
Forever Voyaging 29.95
Bally Hoo 25.95
Cuthroats 25.95
Deadline 31.95
Enchanter 25.95
Hitchiker 25.95
Infidel 28.95
Leather Goddess 25.95
Moonmist 25.95
Planetfall 25.95
Seastalker 25.95
Sorcerer 28.95
Spellbreaker 31.95
Starcross 31.95
Suspect 28.95
Suspended 31.95
Trinity 25.95
Wishbringer 25.95
Witness 25.95
Zork I 25.95
Zork II or III 28.95

ST WORD PROCESSORS
Final Word 89.95
Paperclip Elite 64.95
Habawriter 54.95
Regent Word II 64.95
Wordwriter ST 52.95
Thunder 27.95

HIPPOPOTAMUS
Call for items and prices

MICHTRON
Alt 34.95
BBS ..... Call
Business Tools 34.95
Calendar 20.95
Cornerman 34.95
DOS Shell 27.95
Echo 27.95
Flipside 27.95
Goldrunner 27.95
Kissed 34.95
M-Copy 54.95
M-Disk 27.95
Major Motion 27.95
Mi-Term 34.95
Michtron Utilities 39.95
Mudpies 27.95
Softspool 27.95
Solitaire 27.95
The Animator 27.95
Time Bandits 27.95
Mighty Mail 34.95
Easy Record 54.95
Intro to Logo 34.95
Personal Money Mgr. 34.95
Pinball Factor 27.95
Football Wizard 27.95
Financial Future 27.95
Eight Ball 20.95
Dot Driver 34.95
Laser Driver 34.95
Super Conductor 49.95

ST LANGUAGES
Personal Pascal 49.95
Fast Basic 89.95
Fast C Compiler 99.95
Fast Fortran 199.95
Mark William's C 129.95
Metacomco Pascal 74.95
Macro Assembler 59.95
Lattice C 99.95
Fast Cobol 199.95
MCC Assembler 59.95
Metacomco BCPL 109.95
Cambridge Lisp 139.95
Modula II 54.95
Metacommco Make 49.95
Menu + ... 23.95
BCPL 99.95

ST BUSINESS
VIP Professional 169.95
VIP Lite 99.95
Swiftcalc ST 52.95
Isgur Portfolio 129.95
Synsoft General Ledger 44.95
SBM Point of Sales 84.95
Sierra One Write (ea.) 69.95
Financial Cookbook 34.95
DAC Payroll 34.95
DAC Easy Accounting 49.95
Maxiplan ..... Call
Sylvia Porter Vol. I 52.95
Dollars and Sense 69.95
Home Accountant 44.95
BTS Spreadsheet 44.95

CASIO KEYBOARDS
CZ-101 279
CZ-230S 369

ST EDUCATIONAL
Decimal Dungeon 27.95
Fraction Action 27.95
Kinderama 27.95
Read & Rhyme 27.95
Animal Kingdom 27.95
Teddy Bear/Rainy Day 24.95
Speller Bee 33.95
Kid Talk 33.95
Math Talk 33.95
First Shapes 33.95

FUJI DISKETTES
SS/DD 3.5 in. (10 Pk) ... 15.95
DS/DD 3.5 in. (10 Pk) .. 24.95
NOTE: Buy Fuji Diskettes at these low prices when added to any other order.

ST ACCESSORIES
Flip n' File II-Micro 19.95
Dustcovers ..... Call
3.5 Disk Drive Clean Kit Call
Mouse Pad 8.95
Mouse House 6.95
RCA 6 Way Noise/Surge Prot. ... 34.95
Anti-Glare Screen 19.95

ST UTILITIES
Macrodesk 27.95
Deskmaster 27.95
Music Studio 39.95
Abacus Books ..... Call
Time Link 34.95
ST Music Box 31.95
Middiplay 34.95
Micro-Cookbook 32.95
CZ Droid 74.95
EZ Track 49.95
Right 90 ..... 19.95
Mr. Boston Bartender .. 12.95
Mail List 14.95
Macro-Manager 49.95

EST. 1982
ComputAbility
P.O. Box 17882, Milwaukee, WI 53217
ORDER LINES OPEN
Mon-Fri. 11 a.m. - 7 p.m. CST • Sat. 12 p.m. - 5 p.m. CST

To Order Call Toll Free
800-558-0003
For Technical Info, Order Inquiries, or for Wisc. Orders
414-351-2007

ST ADVENTURES
Hacker 29.95
Hacker II 33.95
9 Princes/Amber 33.95
Sundog 24.95
Winnie The Pooh 19.95
Black Cauldron 27.95
Apshai Trilogy 27.95
Universe II 47.95
The Pawn 29.95
Starquest 33.95
Crimson Crown 16.95
Alternate Reality 33.95
OO-Topos 16.95
Fantacide 33.95
Coveted Mirror 16.95
Golden Pass 29.95
Guild of Thieves 29.95
Dungeon Master 27.95
Tass Times 33.95
Mercenary 27.95
Autoduel 34.95
Ogre 27.95
Gateway 33.95
Defender of Crown 33.95
Sinbad the Sailor 33.95
Balance of Power 33.95
King of Chicago 33.95
S.D.I. 33.95
Ultima II or III 39.95
King's Quest II or III 33.95

ST TELECOMM
PC Intercom 79.95
ST Talk 17.95
I.S. Talk 39.95
Home Pak 33.95

ST PRINT UTILITIES
Typesetter 24.95
Rubber Stamp 24.95
Printmaster 24.95
Art Gallery I 19.95
Art Gallery II 19.95
Fontwriter 27.95
Megafont ST 24.95

ST DATABASES
DB Man 99.95
Regent Base 64.95
Data Manager ST 52.95
DB Master One 39.95
Zoomracks II 99.95

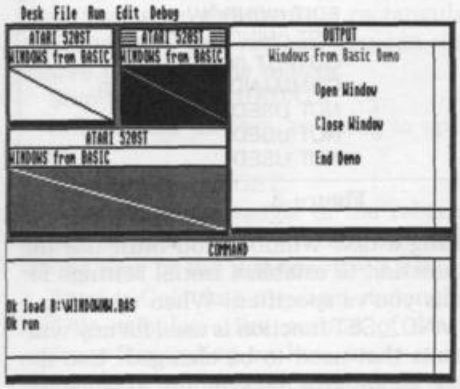
ST ARCADE GAMES
Bridge 4.0 ..... 20.95
Winter Games 27.95
Rogue 27.95
Super Huey 27.95
Phantasie 27.95
Mean 18 ..... 29.95
Leader Board 27.95
Brattacus 33.95
Donald Duck 20.95
Silent Service 27.95
Flight Simulator II 34.95
Champ. Wrestling 27.95
World Games 27.95
Video Vegas 24.95
Blazing Paddles 24.95
Strip Poker 27.95
Starglider 29.95
Alternate Reality 33.95
Little Comp. People 33.95
Smoothtalker ..... Call
Gato 33.95
Joust 20.95
Chessmaster 2000 ..... 32.95
Jet 34.95
Tenth Frame 27.95
Shanghai 29.95
3-D Helicopter 33.95
Micro League Baseball ... 39.95
WWF Micro Wrestling ... 39.95
Skyfox 32.95
Breach 27.95
Star Raiders 20.95
Super Cycle 27.95
ST Pool 24.95
Indoor Sports ..... Call
F-15 Strike Eagle 27.95
High Roller Simulator ..... Call
Two/Two Basketball 29.95
Mastertype 27.95
Hex 24.95
Love Quest 39.95
Deep Space 29.95
Arena 24.95
ST Karate 24.95
ST Protector 23.95
Space Station 24.95
Psion Chess 39.95

ATARI is a trademark of ATARI, INC.

No surcharge for MasterCard or Visa

ORDERING INFORMATION: Please **specify system.** For fast delivery send cashier's check or money order. Personal and company checks allow 14 business days to clear. School P.O.'s welcome. **C.O.D. charges are $3.00.** In Continental U.S.A. include $3.00 for software orders, 4% shipping for hardware. Minimum $4.00. MasterCard and Visa orders please include card #, expiration date and signature. WI residents please include 5% sales tax. HI, AK, FPO, APO, Puerto Rico and Canadian orders, please add 5% shipping, **minimum $5.00.** All other **foreign orders add 15% shipping, minimum $10.00.** All orders shipped outside the Continental U.S.A. are shipped first class insured U.S. mail. If foreign shipping charges exceed the minimum amount, you will be charged the additional amount to get your package to you quickly and safely. All goods are new and include factory warranty. Due to our low prices all sales are final. **All defective returns must have a return authorization number.** Please call (414) 351-2007 to obtain an R.A. # or your return will not be accepted. Prices and availability subject to change without notice.

CIRCLE #107 ON READER SERVICE CARD



Window BASICs
PART 1

An introduction to handling windows in ST BASIC

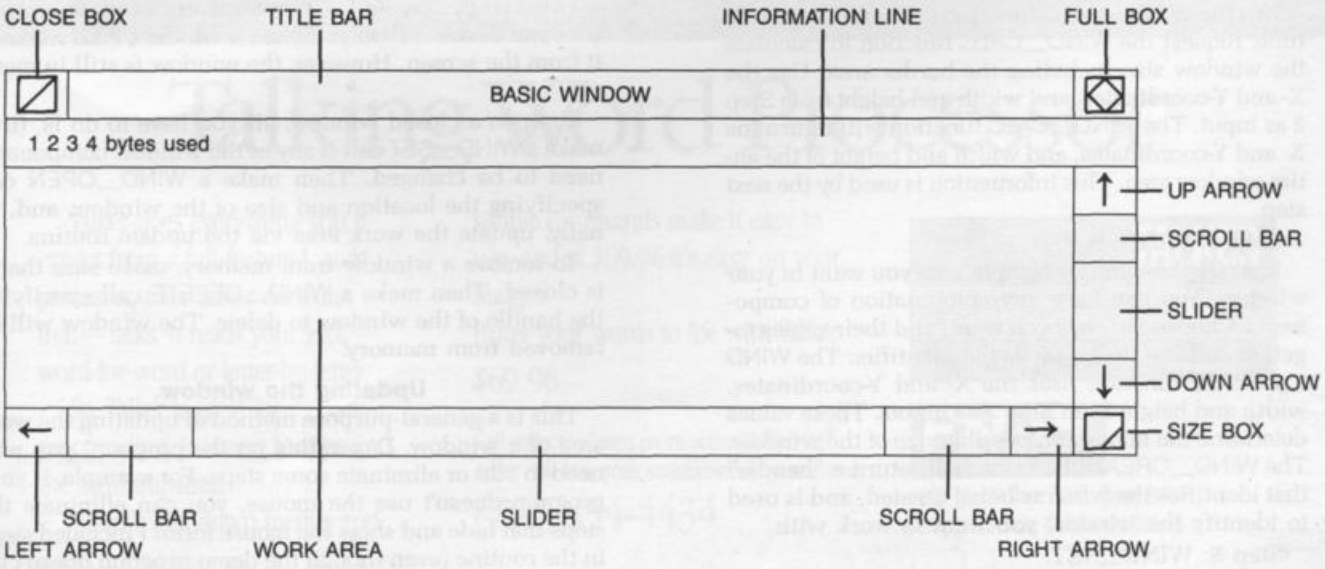
by James Luczak

The GEMSYS() command gives you the ability to create and control "windows" which are independent of those used by ST BASIC. GEM AES supports a maximum of eight windows at a time. The desktop uses one window (the green background behind the BASIC windows), and ST BASIC uses four windows (Edit, List, Output and Command).

This leaves three windows not used by BASIC or the desktop. With the GEMSYS() command, you can create and use up to three windows without disturbing the BASIC environment. And, if you need more than three windows, you can easily delete any or all of the windows used by BASIC, and redesign the windows to meet your specifications.

What exactly is a window? A window is an area of the screen that has clearly defined boundaries, and one or more of the components shown in Figure 1. The area of the screen that displays the disk directory when you double click on a disk icon is an example of a window. The Edit, List, Output and Command areas in ST BASIC are also examples.

Figure 1.



This article is only an introduction to working with windows. Windows can be created using any combination of the components shown in Figure 1. AES will create a window to your specifications, but you're responsible for acting upon the information AES returns to you when your program interacts with a window you've created. If you design a window with many components, you'll have to deal with many different combinations of information returned by AES.

To keep things simple and clear, the windows created in the demo program have only two components: the "title bar" and the "information line." (The "work area" is a part of every window, rather than a component of a window.) To the best of my knowledge, the information contained in this article is correct and complete. I would greatly appreciate any feedback.

Steps for creating a window.

Each step in creating a window is fully commented in the demo program. For specific information about using the GEMSYS() command in creating windows, refer to the demo program. For information about specific GEMSYS() commands, refer to Table 1, at the end of this article.

Step 1: WIND_GET.

Use the WIND_GET function to get information about the desktop window (handle 0). The information returned by the function will identify the area below the menu bar that's available for you to use. The X- and Y-coordinates, along with the width and height of the area, will be returned. This information is used by the next step.

Step 2: WIND_CALC.

Request the WIND_CALC function to calculate a work area for the window you're creating. Use the X- and Y-coordinates, and width and height from Step 1 as inputs. The WIND_CALC function will return the X- and Y-coordinates, width and height of the work area. This information is used by the next step.

Step 3: WIND_CALC.

Use the WIND_CALC function a second time. This time, request the WIND_CALC function to calculate the window size including the border area. Use the X- and Y-coordinates, and width and height from Step 2 as input. The WIND_CALC function will return the X- and Y-coordinates, and width and height of the entire window area. This information is used by the next step.

Step 4: WIND_CREATE.

Identify the window components you want in your window. You can have any combination of components. Choose the ones you want, add their values together and use the result as the identifier. The WIND_CREATE function uses the X- and Y-coordinates, width and height from Step 3 as inputs. These values determine the maximum possible size of the window. The WIND_CREATE function will return a "handle" that identifies the window being created, and is used to identify the window you want to work with.

Step 5: WIND_SET.

WINDOW HANDLES

AES Handle	ST BASIC Window
0	DESKTOP
1	EDIT (WINDOW 0)
2	LIST (WINDOW 1)
3	OUTPUT (WINDOW 2)
4	COMMAND (WINDOW 3)
5	NOT USED
6	NOT USED
7	NOT USED

Figure 2.

When creating a new window, you must use the WIND_SET function to establish initial settings for the components you've specified. When updating a window, the WIND_SET function is used for any window components that need to be changed. Use the window handle to identify the window to work on. Identify the component to change. If more than one component is to be changed, make multiple calls to the WIND_SET function.

Step 6: WIND_OPEN.

Identify the window to open. Input location of window (X- and Y-coordinates of the upper left-hand corner of the window). Input the size of the window (width and height). The WIND_OPEN function places the window at the specified location and displays the window in the size specified. The WIND_OPEN function only displays the window components. It does not update the work area of the window.

Step 7: Update window.

Use the "update routine" (see "Updating the window," below) to write or draw to the work area of the window.

The preceding steps are guidelines for creating windows. It's really a general-purpose method. As you become more familiar with windows, you may find a variation of the steps described to be better suited for your program.

Once you've created a window, you can close or open it as you desire. When you close a window, AES removes it from the screen. However, the window is still in memory.

To open a closed window, all you have to do is, first, make a WIND_SET call if any of the window components need to be changed. Then make a WIND_OPEN call specifying the location and size of the window, and, finally, update the work area via the update routine.

To remove a window from memory, make sure that it is closed. Then make a WIND_DELETE call specifying the handle of the window to delete. The window will be removed from memory.

Updating the window.

This is a general-purpose method of updating the work area of a window. Depending on the program, you may need to add or eliminate some steps. For example, if your program doesn't use the mouse, you can eliminate the steps that hide and show the mouse form. I included steps in the routine (even though the demo program doesn't re-

quire them) to demonstrate the sequence to follow.

Step 1: GRAF_MOUSE.

If you're using the mouse, hide the mouse form. If you don't, and you draw over the mouse form, the next time you move the mouse a rectangular area of the old work area will remain. You can skip this step if you're not using the mouse.

Step 2: WIND_UPDATE.

Inform AES that you're going to update the work area.

Step 3: WIND_GET.

Get the first rectangle of the rectangle list. This function is used when you have overlapping windows in the area of the window that's being updated.

Step 4: Calculate resultant rectangle.

This calculates the portion of the work area to redraw. Depending on how many overlapping windows there are, several rectangular portions of the work area may need to be redrawn. This routine will automatically redraw the correct number of rectangles.

Step 5: Draw or Write routine.

This is where you draw or write to the window that's being updated. In the demo program, the work area is first filled with a background color, then a line is draw diagonally across the area. The first call in the draw routine is to a VDI CLIP function. This function will clip VDI primitives (BAR, CIRCLE, LINE etc.) to the specified size. To PRINT to the window, you must call the VDI TEXT function (see Table 1).

Step 6: WIND_GET.

Use the WIND_GET function to get the next rectangle to update from the rectangle list. This function, along with the draw routine and rectangle calculate routine, is in a loop that will automatically find, calculate and redraw each rectangle that's in the rectangle list.

Step 7: WIND_UPDATE.

Inform AES that you're finished updating the window.

Step 8: GRAF_MOUSE.

Turn the mouse form back on. If your program does not use the mouse, this step can be skipped.

If you want to draw to a window that's already opened and none of the window components need to be changed, simply use this routine to draw to the window. Use the handle to identify the window you want to update.

Running the demo program.

The program should be run in high or medium resolution, with ST BASIC's default windows left unmodified.

WINDOW LIBRARY

CREATE — Establishes full-size window's greatest possible dimensions. Supplies handle.

BASIC CODE	DESCRIPTION
1 a#=gb	
2 gintin=peek(A#+8)	Define Integer Input
3 gintout=peek(a#+12)	Define Integer Output
4 poke gintin,wk	wk=Individual components in the window 1 TITLE BAR NAME 2 CLOSE BOX 4 FULL BOX 8 MOVE BOX 16 INFORMATION LINE 32 SIZE BOX 64 UP ARROW 128 DOWN ARROW 256 VERTICAL SLIDER 512 LEFT ARROW 1024 RIGHT ARROW 2048 HORIZONTAL SLIDER
5 poke gintin+2,wx	wx=Coordinate of full-size window
6 poke gintin+4,wy	wy=Coordinate of full-size window
7 poke gintin+6,ww	ww=Width of full-size window
8 poke gintin+8,wh	wh=Height of full-size window
9 gemsys(100)	OPCODE
10 handle=peek(gintout)	handle=Numeric identifier for this window

NOTE: *wk* — You can create a window with any or all of the components listed. Choose the components you want, add their values together and use the resulting figure for *wk*.
wx,wy,ww,wh — These values are usually derived from the *CALC.* function.
handle — A negative value indicates that no more windows are available.

OPEN — Opens a window in its initial size.

BASIC CODE	DESCRIPTION
1 a#=gb	
2 gintin=peek(a#+8)	Define Integer Input
3 gintout=peek(a#+12)	Define Integer Output
4 poke gintin,handle	handle=Identifier of window to be opened Supplied by the CREATE function
5 poke gintin+2,wx	wx=Coordinate of window (initial size)
6 poke gintin+4,wy	wy=Coordinate of window (initial size)
7 poke gintin+6,ww	ww=Width of window (initial size)
8 poke gintin+8,wh	wh=Height of window (initial size)
9 gemsys(101)	OPCODE
10 return=peek(gintout)	return=Status message 0 ERROR Positive Integer NO ERROR

NOTE: This function determines the initial size and placement of the window.

DELETE — Frees the space occupied by a window.

BASIC CODE	DESCRIPTION
1 a#=gb	
2 gintin=peek(a#+8)	Define Integer Input
3 gintout=peek(a#+12)	Define Integer output
4 poke gintin,handle	handle=Identifier of window to be deleted.
5 gemsys(103)	OPCODE
6 return=peek(gintout)	return=Status message 0 ERROR Positive Integer NO ERROR

GET — Returns various information about a window.

BASIC CODE	DESCRIPTION
1 a#=gb	
2 gintin=peek(a#+8)	Define Integer Input
3 gintout=peek(a#+12)	Define Integer Output
4 poke gintin,handle	handle=Identifier of window requesting information
5 poke gintin+2,field	field=Identifies field of information 4 Coordinates of work area 5 Coordinates of entire window, including title bar, borders, etc. 6 Coordinates of the entire previous window, including title bar, borders, etc. 7 Coordinates of window at its largest size, including title bar, borders, etc. 8 Relative position of horizontal slider (will be a number between 1 and 1000) 9 Relative position of vertical slider (will be a number between 1 and 1000) 10 Handle of the window that is active 11 Coordinate of the first rectangle in the window's rectangle list 12 Coordinate of the next rectangle in the window's rectangle list 15 Size of the horizontal slider 16 Size of the vertical slider
6 gemsys(104)	OPCODE
7 return=peek(gintout)	return=Status message 0 ERROR Positive Integer NO ERROR
8 w1=peek(gintout+2)	w1=see note below
9 w2=peek(gintout+4)	w2=see note below
10 w3=peek(gintout+6)	w3=see note below
11 w4=peek(gintout+8)	w4=see note below

NOTE: The information returned in *w1,w2,w3,w4* depends on the FIELD value.

FIELD VALUE	INFORMATION RETURNED IN w1-w4
4	w1=X-coordinate w2=Y-coordinate w3=Width w4=Height
5	w1=X-coordinate w2=Y-coordinate w3=Width w4=Height
6	w1=X-coordinate w2=Y-coordinate w3=Width w4=Height
7	w1=X-coordinate w2=Y-coordinate w3=Width w4=Height
8	w1=1 — LEFTmost position 1000 — RIGHTmost w2=not used w3=not used w4=not used
9	w1=1 — TOP position 1000 — BOTTOM w2=not used w3=not used w4=not used
10	w1=Handle of the active window. w2=not used w3=not used w4=not used
11	w1=X-coordinate w2=Y-coordinate w3=Width w4=Height
12	w1=X-coordinate w2=Y-coordinate w3=Width 4=Height
15	w1=−1 — Default minimum size (square box) 1-1000 — Slider's relative size compared to the horizontal scroll bar w2=not used w3=not used w4=not used
16	w1=−1 — Default minimum size (square box) 1-1000 — Slider's relative size compared to the vertical scroll bar w2=not used w3=not used w4=not used

SET — Changes values in various window display fields.

BASIC CODE	DESCRIPTION
1 a#=gb	
2 gintin=peek(a#+8)	Define Integer Input
3 gintout=peek (a#+12)	Define Integer Output
4 poke gintin,handle	handle=Identifier of window requesting change
5 poke gintin+2,field	field=Identifies field to change 1 Window components 2 Address of string containing title bar data 3 Address of string containing information line data 5 Coordinates of the entire window, including title bar, borders, etc. 8 Relative position of horizontal slider 9 Relative position of vertical slider 10 Handle of window that is active 14 Address of a new default desktop 15 Size of horizontal slider 16 Size of vertical slider
6 poke gintin+4,w1	w1=see note below
7 poke gintin+6,w2	w2=see note below
8 poke gintin+8,w3	w3=see note below
9 poke gintin+10,w4	w4=see note below
10 gemsys(105)	OPCODE
11 return=peek (gintout)	return=Status Message 0 ERROR Positive Integer NO ERROR

NOTE: how to calculate low-high addresses —

BASIC CODE	
1 a$="hello"	
2 addr=varptr(a$)	See the *ST BASIC Sourcebook* for more information about the VARPTR command.
3 hiaddr=int(addr/65536)	
4 loaddr=addr−(hiaddr*65536)	

The information required by *w1,w2,w3,w4* depends on the FIELD value.

FIELD VALUE	INFORMATION REQUIRED IN w1-w4
1	w1= 1 TITLE BAR NAME 2 CLOSE BOX 4 FULL BOX 8 MOVE BOX 16 INFORMATION LINE 32 SIZE BOX 64 UP ARROW 128 DOWN ARROW 256 VERTICAL SLIDER 512 LEFT ARROW 1024 RIGHT ARROW 2048 HORIZONTAL SLIDER w2=not used w3=not used w4=not used

You can change window parameters with any or all the components listed. Choose the components you want, add their values together and use the resulting figure.

2	w1=Hiaddr w2=Loaddr w3=not used w4=not used	5	w1=X-coordinate w2=Y-coordinate w3=Width w4=Height
3	w1=Hiaddr w2=Loaddr w3=not used w4=not used	8	w1=1 — LEFTmost position 1000 — RIGHTmost w2=not used w3=not used w4=not used

SET continued

```
 9   w1=1 — TOP position
         1000 — BOTTOM
     w2=not used
     w3=not used
     w4=not used
10   w1=Handle of the active window
     w2=not used
     w3=not used
     w4=not used
14   w1=Low word (object tree)
     w2=High word (object tree)
     w3=Starting object in object tree
     w4=not used
15   w1=-1 — Default minimum size
          (square box)
     1-1000 — Slider's relative
          size compared to the
          horizontal scroll bar
     w2=not used
     w3=not used
     w4=not used
16   w1=-1 — Default minimum size
          (square box)
     1-1000 — Slider's relative
          size compared to the
          vertical scroll bar
     w2=not used
     w3=not used
     w4=not used
```

FIND — Finds what window is under the mouse's X,Y-coordinates.

BASIC CODE	DESCRIPTION
1 a#=gb	
2 gintin=peek(A#+8)	Define Integer Input
3 gintout=peek(a#+12)	Define Integer Output
4 poke gintin,mx	mx=Coordinate of mouse
5 poke gintin+2,my	my=Coordinate of mouse
6 gemsys(106)	OPCODE
7 handle=peek(gintout)	handle=Identifier of window under the mouse coordinates

UPDATE — Notifies AES that you are about to

(a) — End Update a window
(b) — Begin Updating a window
(c) — End Mouse control functions
(d) — Begin Mouse control functions

BASIC CODE	DESCRIPTION
1 a#+gb	
2 gintin=peek(A#=8)	Define Integer Input
3 gintout=peek(a#+12)	Define Integer Output
4 poke gintin,fc	fc=Function call 0 End Update 1 Begin Update 2 End mouse control 3 Begin mouse control
5 gemsys(107)	OPCODE
6 return=peek(gintout)	return=Status Message 0 ERROR Positive Integer NO ERROR

CALC — Calculates X, Y, width, height of window's border area and work area.

BASIC CODE	DESCRIPTION
1 a#=gb	
2 gintin=peek(A#+8)	Define Integer Input
3 gintout=peek(a#+12)	Define Integer Output
4 poke gintin,t	t=type of calculation 0 Border area 1 Work area
5 poke gintin+2,k	k=Components present in the window 1 TITLE BAR NAME 2 CLOSE BOX 4 FULL BOX 8 MOVE BOX 16 INFORMATION LINE 32 SIZE BOX 64 UP ARROW 128 DOWN ARROW 256 VERTICAL SLIDER 512 LEFT ARROW 1024 RIGHT ARROW 2048 HORIZONTAL SLIDER
6 poke gintin+4,ix	ix=see note below
7 poke gintin+6,iy	iy=see note below
8 poke gintin+8,iw	iw=see note below
9 poke gintin+10,ih	ih=see note below
10 gemsys(108)	OPCODE
11 return=peek(gintout)	return=Status Message 0 ERROR Positive Integer NO ERROR
12 ox=peek(gintout+2)	ox=see note below
13 oy=peek(gintout+4)	oy=see note below
14 ow=peek(gintout+6)	ow=see note below
15 oh=peek(gintout+8)	oh=see note below

NOTE: The INPUTS *ix, iy, iw, ih*, and the OUTPUTS *ox, oy, ow, oh* depend on what calculation is being performed.

CALCULATION TYPE	PARAMETERS
0 - BORDER AREA	
INPUTS	ix=X-coordinate of work area iy=Y-coordinate of work area iw=Width of work area ih=Height of work area
OUTPUTS	ox=X-coordinate of border area oy=Y-coordinate of border area ow=Width of border area oh=Height of border area
1 - WORK AREA	
INPUTS	ix=X-coordinate of border area iy=Y-coordinate of border area iw=Width of border area ih=Height of border area
OUTPUTS	ox=X-coordinate of work area oy=Y-coordinate of work area ow=Width of work area oh=Height of work area

VDI CALLS

SET CLIPPING RECTANGLE

BASIC	CODE DESCRIPTION
1 poke contrl,129	OPCODE
2 poke contrl+2,2	
3 poke contrl+6,1	
4 poke intin,fl	fl=Clipping Indicator 0 Clipping OFF 1 Clipping ON
5 poke ptsin,x	x=Coordinate of clipping rectangle
6 poke ptsin+2,y	y=Coordinate of clipping rectangle
7 poke ptsin+4,x1	x1=Coordinate diagonally across from ptsin
8 poke ptsin+6,y1	y1=Coordinate diagonally across from ptsin+2
9 vidsys(1)	

NOTE: This function enables or disables clipping. If clipping is on, all primitives (BAR, CIRCLE, ELLIPSE, etc.) are clipped to the size specified by this function.

TEXT

BASIC CODE	DESCRIPTION
1 poke contrl,8	OPCODE
2 poke contrl+2,1	
3 poke contrl+6,num	num=Number of characters to display
4 poke intin,char1	char1=1st character to display (ASCII value)
5 poke intin+2,char2	char2=Next character to display
6 poke intin+n,charn	charn=Last character to display
7 poke ptsin,x	x=Coordinate to display text at
8 poke ptsin+2,y	y=Coordinate to display text at
9 vdisys(1)	

NOTE: This function can be used to display text at any X- or Y-coordinates on the screen. The text must be entered as ASCII values.

EXAMPLE: Display the word *HELLO*

poke contrl+6,5	(Length of word to be displayed)
poke intin,72	(H)
poke intin+2,69	(E)
poke intin+4,76	(L)
poke intin+6,76	(L)
poke intin+8,79	(O)

The demo program demonstrates how you can create, open, update, close and delete windows from BASIC. A short menu will appear in the BASIC output window. By pressing the letter *O*, you can open up to three windows. These will appear in the area of the screen where the BASIC list window usually is. Each time you press *O*, a window will appear in this area. If you already have three windows open, the program will cycle back to the first window.

To close a window, press the letter *C*. Each time you do so, a window that's on the screen will be removed. If all windows are closed, nothing will happen.

To exit the program, press *E*. You can exit the program at any time. If there are any windows still open, they will be closed. When you exit the program, all windows that have been created are closed and deleted from memory.

As I mentioned earlier, this article just scratches the surface of AES windowing capabilities. Everything you see windows used for on the desktop (and in other programs) can be accomplished from BASIC. It's not hard to work with windows—if you take a step-by-step approach.

Caution: Make sure to save your program as you experiment with creating and manipulating your own windows. Usually, if you make a mistake, the system will crash—

and you'll lose your program.

Next month, we'll continue our exploration of windows through ST BASIC.

Listing 1.
ST BASIC listing.

```
100 '* WINDOWS FROM BASIC DEMO *
110 '****** by JIM LUCZAK*******
120 a#=gb:gintin=peek(a#+8):gintout=pe
ek(a#+12)
150 a$=" ATARI 520ST ":aaddr=varptr(a$
)
160 aadhi=int(aaddr/65536):aadlo=aaddr
-(aadhi*65536)
170 b$="WINDOWS from BASIC":baddr=varp
tr(b$)
180 badhi=int(baddr/65536):badlo=baddr
-(badhi*65536)
190 closew 0:closew 1:clearw 2
200 color 2,1,1:?"       Windows From
Basic Demo":?
210 color 2,1,1:?"                  O";:c
olor 1,1,1:?"pen Window":?
220 color 2,1,1:?"                  C";:c
olor 1,1,1:?"lose Window":?
230 color 2,1,1:?"                  E";:c
olor 1,1,1:?"nd Demo":?
240 fc=1:poke systab+24,1
250 '------- MAIN PROGRAM LOOP ------
260 while mc=0
270 gosub GETKEYPRESS
280 if kb=6223 or kb=6255 then gosub D
OCREATE:try=0
290 if kb=11843 or kb=11875 then gosub
 WCLOSE:gosub CHKHAND
300 wend
310 goto CLEANUP
320 '- STEPS FOR CREATING A WINDOW -
330 DOCREATE:
340 if crt=1 then goto CHKHAND1
350 '---------- WIND_GET ----------
360 poke gintin,0:' Get info on Deskto
p Window (0)
370 poke gintin+2,4:' Get Avaiable are
a coordinates
380 gemsys(104)
390 wx1=peek(gintout+2):' X coordinate
  Use as Input in WIND_CALC
400 wy1=peek(gintout+4):' Y coordinate
  Use as Input in WIND_CALC
410 ww1=peek(gintout+6):' Width, Use a
s Input in WIND_CALC
420 wh1=peek(gintout+8):' Height, Use
as Input in WIND_CALC
430 '--------- WIND_CALC ----------
440 poke gintin,1:' Calculate Work are
a
450 poke gintin+2,17:' Identify window
 components to use
460 poke gintin+4,wx1:' X coordinate f
rom WIND_GET
470 poke gintin+6,wy1:' Y coordinate f
rom WIND_GET
480 poke gintin+8,ww1:' Width  from WI
ND_GET
490 poke gintin+10,wh1:' Height from W
IND_GET
500 gemsys(108)
510 wx2=peek(gintout+2):' X coordinate
 Use as Input in WIND_CALC 2
520 wy2=peek(gintout+4):' Y coordinate
 Use as Input in WIND_CALC 2
530 ww2=peek(gintout+6):' Width Use as
 Input in WIND_CALC 2
540 wh2=peek(gintout+8):' Height Use a
s Input in WIND_CALC 2
550 '------- WIND_CALC 2 ---------
560 poke gintin,0:' Calc window size i
ncluding Border area
570 poke gintin+2,17:' Identify window
 components to use
580 poke gintin+4,wx2:' X coordinate f
rom WIND_CALC
590 poke gintin+6,wy2:' Y coordinate f
rom WIND_CALC
600 poke gintin+8,ww2:' Width from WIN
D_CALC
610 poke gintin+10,wh2:' Height from W
IND_CALC
620 gemsys(108)
630 wx3=peek(gintout+2):' X coordinate
 Use as Input in WIND_CREATE
640 wy3=peek(gintout+4):' Y coordinate
 Use as Input in WIND_CREATE
650 ww3=peek(gintout+6):' Width Use as
 Input in WIND_CREATE
660 wh3=peek(gintout+8):' Height Use a
s Input in WIND_CREATE
670 '------- WIND_CREATE ---------
680 poke gintin,17:' Identify window c
omponents to use
690 poke gintin+2,wx3:' X coordinate f
rom WIND_CALC 2
700 poke gintin+4,wy3:' Y coordinate f
rom WIND_CALC 2
710 poke gintin+6,ww3:' Width from WIN
D_CALC 2
720 poke gintin+8,wh3:' Height from WI
ND_CALC 2
730 gemsys(100)
740 handle=peek(gintout):' HANDLE of w
indow being created
750 if handle=7 then crt=1
760 hand=handle:gosub WLOCATION
780 '---------- WIND_SET ----------
790 WINDSET:
800 poke gintin,handle:' Identify Wind
ow
810 poke gintin+2,2:' Change TITLE lin
e
820 poke gintin+4,aadhi:' High word ad
dress of Title string.
830 poke gintin+6,aadlo:' Low word add
ress Title string.
840 poke gintin+8,0:' Not Used
850 poke gintin+10,0:' Not Used
860 gemsys(105)
870 poke gintin+2,3:' Change INFORMATI
ON line
880 poke gintin+4,badhi:' High word ad
dress of Information string
890 poke gintin+6,badlo:' Low word add
ress of Information string
900 poke gintin+8,0:' Not Used
910 poke gintin+10,0:' Not Used
920 gemsys(105)
930 '--------- WIND_OPEN ----------
940 poke gintin,handle:' Identify Wind
ow
950 poke gintin+2,ax:' X coordinate (l
ocation to open window)
960 poke gintin+4,ay:' Y coordinate (l
ocation to open window)
970 poke gintin+6,aw:' Width of window
980 poke gintin+8,ah:' Height of windo
w
990 gemsys(101)
1000 '----- UPDATE WINDOW -------
1010 '
1020 '--- DRAW LINE IN WINDOW ---
1030 gosub UPDATER:return
1050 '----- CLEAN-UP AND END -----
1060 CLEANUP:
1070 poke systab+24,0
```

```
1080 for hand=5 to 7:gosub WCLOSE:gosu
b WDELETE:next hand
1120 clearw 2:end
1130 '--- WINDOW UPDATE ROUTINE ---
1140 UPDATER:
1150 mf=256:gosub GMOUSE:' Hide Mouse
form
1160 udv=1:gosub WUPDATE:' Begin Updat
e
1170 getv=11:gosub WGET:' Get First up
date rectangle
1180 while rw>0 and rh>0
1190 gosub RESULT:' Calculate rectangl
e parameters
1200 gosub DROUTINE:' Draw or write to
 window being updated
1210 getv=12:gosub WGET:' Get Next upd
ate rectangle
1220 wend
1230 udv=0:gosub WUPDATE:' End Update
1240 mf=257:gosub GMOUSE:' Show Mouse
form
1250 return
1260 ' CALCULATE RESULTANT RECTANGLE
1270 RESULT:
1280 x1=wx1:y1=wy1:w1=ww1:h1=wh1
1290 if x1+w1<rx+rw then triw=x1+w1 el
se triw=rx+rw
1300 if y1+h1<ry+rh then trih=y1+h1 el
se trih=ry+rh
1310 if x1>rx then trix=x1 else trix=r
x
1320 if y1>ry then triy=y1 else triy=r
y
1330 triw=triw-trix:trih=trih-triy
1350 return
1360 '--------- GRAF_MOUSE ----------
1370 GMOUSE:
1380 poke gintin,mf:' Mouse Form
1390 gemsys(78):return
1410 '-------- WIND_UPDATE -------
1420 WUPDATE:
1430 poke gintin,udv:' Begin / End Upd
ate
1440 gemsys(107):return
1460 '--------- WIND_GET -----------
1470 WGET:
1480 poke gintin,handle:' Identify Win
dow
1490 poke gintin+2,getv:' Identify inf
ormation request
1500 gemsys(104)
1510 rx=peek(gintout+2):' X coordinate
 of rectangle
1520 ry=peek(gintout+4):' Y coordinate
 of rectangle
1530 rw=peek(gintout+6):' Width of rec
tangle
1540 rh=peek(gintout+8):' Height of re
ctangle
1550 return
1560 '------- CLOSE WINDOW --------
1570 WCLOSE:
1580 if h(hand-4)<>1 then return
1590 poke gintin,hand:' Identify windo
w
1600 gemsys(102):try=1:h(hand-4)=0
1620 return
1630 '------ DELETE WINDOW ---------
1640 WDELETE:
1650 if wd(hand-4)<>1 then return
1660 poke gintin,hand:' Identify windo
w
1670 gemsys(103):return
1690 '------- DRAW ROUTINE -------
1700 DROUTINE:
1710 bc=bc+1:if bc>3 then bc=0
1720 fc=fc+1:if fc>3 then fc=0
1730 color 1,bc,fc:' Do Window Fill an
d Line colors
1740 '.. CLIP TO RECTANGLE SIZE ..
1750 poke contrl,129:' OPCODE
1760 poke contrl+2,2:poke contrl+6,1
1780 poke intin,1:' Turn clipping ON
1790 poke ptsin,trix:' X coordinate to
 clip to
1800 poke ptsin+2,triy:' Y coordinate
to clip to
1810 poke ptsin+4,trix+triw:' X coordi
nate diagonally across (ptsin)
1820 poke ptsin+6,triy+trih:' Y coordi
nate diagonally across (ptsin+2)
1830 vdisys(1)
1840 '.... FILL RECTANGLE ....
1850 poke contrl,114:' OPCODE
1860 poke contrl+2,2:poke contrl+6,0
1880 poke ptsin,trix:' X coordinate of
 rectangle
1890 poke ptsin+2,triy:' Y coordinate
of rectangle
1900 poke ptsin+4,trix+triw:' X coordi
nate diagonally across (ptsin)
1910 poke ptsin+6,triy+trih:' Y coordi
nate diagonally across (ptsin+2)
1920 vdisys(1)
1930 '..... DRAW LINE .....
```

(Listings continue on page 24)

ST news

Publishing Partner now available

Soft Logik Corp. now produces a full-featured desktop publishing system, with complete screen representation of your printed page. **Publishing Partner** is designed for those requiring high-quality output consisting of both text and graphics, on dot-matrix or laser printers. Popular dot-matrix and any Postscript-compatible printers are supported—including LaserWriter printers from Apple.

Publishing Partner permits you to adjust the character size and spacing, line spacing and, with the "toolbox" option, lets you insert lines, boxes, circles and patterns—anywhere on the page. Word processing, page layout and forms creator fill out the package. **Partner**'s additional features include: multicolumn format, highlighting, multiple-font capability, and much more.

Retail is $149.95. For more information, write or call Soft Logik Corporation, 4129 Old Baumgartner, St. Louis, MO 63129 — (314) 894-8608. Reader Service #121.

Latest Abacus book

Number eight in the popular series of ST publications from Abacus is ***Peeks & Pokes***. This 175-

page book provides all the information you need to customize a desktop, change character set or mouse cursor shape, read joystick port and keyboard, plus how to alter fill patterns, get direct disk access, and more. $16.95, by Stefan Dittrich, translated from German. ISBN #0-916439-56-9. Abacus Software, P.O. Box 7219, Grand Rapids, MI 49510 — (616) 241-5510. Reader Service #124.

ST-Base on-line

1ST Base Software calls their new **ST-Base** a full-featured BBS at an affordable price, with a full range of features. Up to sixteen special interest groups, sixteen file transfer areas, user-written story board (with full "bad words" dictionary), on-line questionnaire, screen protect feature and a sophisticated message center are only a few of the options.

Also offered: an 80-column graphic display of BBS statistics, with hourly usage and most popular downloaded files. Full Xmodem protocol compatible. A 41-page manual is provided. **ST-Base** costs $50.00 ($70.00 in Canada). 1ST Base Software, 48 Amherst Crescent, Nepean, Ontario, K2J 1V9, Canada — BBS (613) 231-3411. Reader Service #123.

Universal MIDI librarian

Key Clique has announced a universal MIDI librarian package that allows musicians to store songs and sound to disk. **SYS/EX** has been out for some time on the Apple II, Commodore 64 and IBM PC, and now makes its debut on the ST. It's compatible with nearly sixty synthesizers, samplers, drum machines and other MIDI instruments. A demo disk is available for $5.00. For more on **SYS/EX**, contact Key Clique at 3960 Laurel Canyon Blvd., Suite 374, Studio City, CA 91604 — (818) 905-9136. Reader Service #125.

Starglider flying

Now shipping from Firebird is their "3-D vector graphic" game, **Starglider**. An animated, high-speed combat simulator, it puts the player in the role of a pilot opposing the alien invaders from the planet Novenia. The mission goal is to destroy the flagship

Starglider, using flying skill and battle strategy. A novella accompanies the game, providing clues needed to survive the battle and its increasing levels of difficulty. Firebird also distributed **The Pawn**. Priced at $39.95. Firebird Licensees, Inc., P.O. Box 49, Ramsey, NJ 07446. Reader Service #122.

Four new titles from MichTron

Back up your hard disk to floppies, quickly and easily, with **Backup!** This utility uses GEM drop-down menus to make things easier. Several types of backups are available, as well as numerous options, including the ability to copy only newly created files, the entire disk contents, or files by date only. $39.95.

Pinball Factory is a pinball game with a plus. As the title implies, the user can design and "build" his own screen, save it, then play it. Change the logo, select from bumpers, walls, tabs, and more. Commands let you draw lines, round or squared-off frames, airbrush, etc. Up to 16 colors can be chosen from the ST's palette of over 500. Gravity, bounce, scoring, bumper strength and tab bonuses can be altered, and up to four can play. $39.95.

Another new game, **Eight Ball**, gives you a realistic overhead view of a pool table, with everything drawn to regulation proportions. A game for one or two players, it runs on monochrome or color systems. $29.95.

Back to serious software. . .**Your Financial Future** is more easily planned by taking advantage of this program. Projections, net worth, investments, annual savings, inflation and investment yields can all be analyzed, for $39.95.

All from MichTron, 576 S. Telegraph, Pontiac, MI 48053 — (313) 334-5700. Reader Service #126.

Other news

☑ Micro-W is now offering QRS's **Piano Roll** library—an extensive (over 10,000-song) music collection in a wide variety of styles, early ragtime to rock. Each 6-song album is $19.95, listed in a catalog from Micro-W Distributing, Inc., Butler, NJ 07405. Reader Service #127.

☑ **Dollars and Sense** runs on any ST and fully supports a hard disk. Double-entry accounting, detailed financial statements, check printing, graph generating and financial planning are only *some* of its capabilities. $99.95. Monogram, 8295 S. LaCienega Blvd., Inglewood, CA 90301 — (213) 215-0355. Reader Service #128.

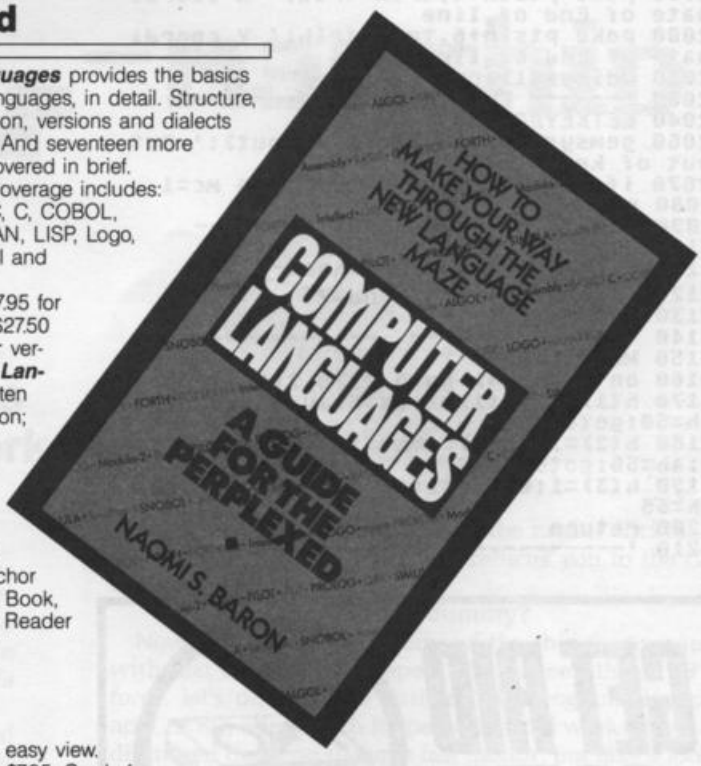
☑ Touted as the all-in-one electronic GEM desktop organizer, **Inagem Agenda** keeps multiple records of your past, present and future events. With simple mouse movements, this program acts as a calendar, diary, phone book and reminder alarm of unlimited capacity. A high-speed search and retrieval system is built in, plus the ability to store up to 3200 characters a day, and print out memo and phone book hard copy. $49.95. Inagem Technologies, Inc., 6177 Gerard Morisset Ave., Montreal, Quebec H1M 3J8, Canada — (514) 256-9942. Reader Service #129.

A guide for the perplexed

Computer Languages provides the basics on twenty-two languages, in detail. Structure, genealogy, function, versions and dialects are all explored. And seventeen more languages are covered in brief.

The in-depth coverage includes: Assembly, BASIC, C, COBOL, FORTH, FORTRAN, LISP, Logo, Modula-2, Pascal and PILOT.

The cost is $17.95 for the paperback, $27.50 for the hardcover version. ***Computer Languages*** was written by Naomi S. Baron; 419 pages of information. To order, ask your book dealer for ISBN #0-385-23213-6. Published by Anchor Press/Doubleday Book, Garden City, NY. Reader Service #131.



☑ **QuikCards** are quick-reference cards designed to slip between function keys and the computer case, for easy view. The latest set supports the telecommunications program **Flash**, for $7.95. Cards for **ST-Writer** and **ST-Talk** are available as a set, for $5.95. From Hired Hand Graphics, 1010 NE Dewey Drive, Grants Pass, OR 97526 — (503) 476-6931. Reader Service #130.

Talk to me

Four educational programs have just been released from First Byte with self-contained, unlimited text-to-speech capabilities. For this, the programs employ "Smooth-talker" speech technology developed by First Byte. The ST's color and graphics are put to use, as well as its audio functions, to provide exciting interactive entertainment.

Speller Bee improves children's spelling skills by giving them a variety of games and simulated test situations. **Kidtalk** is a talking word processor for kids, which helps to up writing skills and reading abilities. Math functions are covered in **Mathtalk**, where addition, subtraction, multiplication and division are learned.

Finally, Ted E. Bear is a talking friend for the young user, in **First Shapes**. The goal here is to develop interest in mathematics and to improve reading, writing and problem solving skills.

For more information on these programs, call First Byte: (213) 595-7006, 8:30 am to 4:30 pm PST, M-F. First Byte, Inc., 2845 Temple Avenue, Long Beach, CA 90806. Reader Service #132.

Transform

. . . is a modular music system which provides all of the features used in professional music production today. The system consists of three modules, the first of which is **Xtrack**, a a musical sequencer and recorder. Using **Xtrack**, an unlimited number of tracks can be recorded on, named and edited. A MIDI event editor, for logical graphic and grid mode, is also included. The program is $149.95.

Xnotes is a musical composing and arranging program, with windows, icons and pull-down menus. Use the mouse to click-on standard musical notation. You can also print out high-resolution, hard copy musical scores, all for $199.95.

The third program in the series, **Xsyn**, allows you to edit music for quick sound alterations. With the built-in sound creator, you can generate new sounds instantly. An included real-time recorder is set up with more than 50,000 notes. This program is selling for $99.95.

Combine the modules for a complete sound system. From Beam Team, 6100 Adeline Street, Oakland, CA 94608 — (415) 658-3208. Reader Service #133.

```
1940 poke contrl,6:' OPCODE
1950 poke contrl+2,2:poke contrl+6,0
1970 poke ptsin,trix:' X coordinate of
 Start of line
1980 poke ptsin+2,triy:' Y coordinate
of Start of line
1990 poke ptsin+4,trix+triw:' X coordi
nate of End of line
2000 poke ptsin+6,triy+trih:' Y coordi
nate of End of line
2010 vdisys(1):return
2030 '------ EVNT_KEYBOARD --------
2040 GETKEYPRESS:
2060 gemsys(20):kb=peek(gintout):' Out
put of keypress
2070 if kb=4709 or kb=4677 then mc=1
2080 return
2090 '----- MISC ROUTINES --------
2100 CHKHAND:
2110 if try=1 then hand=hand-1
2120 if hand<5 then hand=7
2130 return
2140 '----------------------------
2150 WLOCATION:
2160 on (hand-4) goto 2170,2180,2190
2170 h(1)=1:wd(1)=1:ax=5:ay=12:aw=150:
ah=50:goto 2200
2180 h(2)=1:wd(2)=1:ax=165:ay=12:aw=15
0:ah=50:goto 2200
2190 h(3)=1:wd(3)=1:ax=5:ay=65:aw=310:
ah=55
2200 return
2210 '----------------------------
2220 CHKHAND1:
2230 handle=handle+1:if handle>7 then
handle=5
2240 hand=handle
2250 if h(hand-4)=1 then gosub WCLOSE
2260 gosub WLOCATION:goto WINDSET
```

●

ST CHECKSUM DATA.

(see page 84)

```
 100 data  974, 511, 932, 803, 317, 5,
 334, 768, 967, 227, 5838
 220 data  591, 599, 41, 188, 778, 934
, 463, 654, 48, 130, 4426
 320 data  420, 666, 953, 685, 361, 56
1, 784, 487, 468, 545, 5930
 420 data  655, 721, 53, 410, 858, 872
, 810, 890, 776, 615, 6660
 520 data  624, 869, 792, 491, 631, 41
8, 898, 912, 960, 978, 7573
 620 data  784, 840, 849, 51, 17, 714,
 457, 112, 98, 92, 4014
 720 data  229, 765, 510, 352, 278, 72
6, 618, 399, 899, 777, 5553
 830 data  496, 766, 884, 791, 823, 65
5, 609, 756, 874, 781, 7435
 930 data  808, 413, 113, 126, 948, 14
1, 790, 669, 522, 830, 5360
 1030 data  339, 753, 586, 501, 631, 1
38, 280, 576, 356, 126, 4286
 1170 data  996, 772, 832, 64, 715, 17
, 765, 427, 448, 747, 5783
 1270 data  511, 108, 435, 285, 958, 9
69, 860, 451, 921, 476, 5974
 1380 data  190, 901, 774, 628, 118, 9
96, 689, 137, 427, 169, 5029
 1500 data  863, 686, 694, 595, 841, 4
57, 792, 463, 902, 196, 6489
 1600 data  108, 457, 892, 585, 44, 19
6, 1, 697, 834, 444, 4258
 1720 data  469, 551, 15, 572, 475, 55
1, 85, 221, 64, 198, 3201
 1830 data  721, 364, 558, 477, 435, 5
93, 66, 200, 723, 157, 4294
 1940 data  378, 479, 37, 928, 290, 25
1, 704, 971, 303, 229, 4570
 2070 data  78, 447, 710, 543, 583, 10
9, 445, 185, 940, 801, 4841
 2170 data  98, 409, 902, 445, 50, 704
, 852, 975, 482, 444, 5361
```

●

C-manship

A study of VDI text functions and getting programs to work in any resolution.

by Clayton Walnum

Those of you who programmed the 8-bit Ataris were limited in your text displays. Sure, you had graphics 1 and 2, which endowed your computer with oversized text in four colors, and you could, when in graphics 0, inject life with some inverse video.

If those alternatives did nothing to satisfy your critical eye, you could always take refuge in a redesigned character set. And, if you were into self-brutalization—or were desperate to the point where opened wrists seemed preferable to another moment of programming—you could draw your characters pixel by pixel, line by line, until your masterwork emerged amidst the ruins of your mental health.

But those are bygone times. Now you own an ST. Because the ST's screen is bit-mapped rather than character-mapped, you may fire your shrink and discard all schemes of self-destruction. Text, like any other graphic, is drawn on the screen.

Stop right there! Wasn't it the *drawing* of text on the 8-bits—that ghastly alternative to the normal displays—that forced many talented bit-and-byte managers to take up residence in the local Institute for the Incredibly Nervous? Yes, indeed. But, on the ST, GEM's VDI takes on the task, supplying the programmer with simple functions to graphically manipulate text. There are about two dozen text sizes available, as well as numerous special effects, which can be combined in any way the programmer sees fit.

To get a quick introduction to the VDI text functions, type in Listing 1, compile it (it was written with Megamax C; if you own a different compiler, you may have to make some changes) and run it. Use the mouse to click on the menu options. Clicking the left button when viewing a demo screen returns you to the menu; clicking the right button when at the menu returns you to the GEM desktop.

Who's a dummy?

Now that you've seen some of the things you can do with text on an ST (I suspect you've seen this stuff before), let's dig into the listing. The program first calls appl_init(), after which it opens a virtual workstation. We discussed these procedures last month, but take a look at the parameters for the graf_handle() call. See something a little strange? Four of the parameters are the address of the variable dummy.

Last month, I told you that graf_handle() returns information about the system font. This information is stored in four variables whose addresses you pass with the call. In this month's demo program, we've no need for this information, so why clutter up the program with extra variables? The graf_handle() call doesn't care where it stores the information, as long as you give it an address. In fact, it doesn't even care if you give it the same address for all four values. It'll happily store one value on top of the previous one (wiping the older value out, of course; you'll have no way to retrieve any but the last).

The integer variable dummy is used throughout the program in just this way. Anytime we must supply storage for a dispensable value, we'll use the dummy variable.

Converting between resolutions.

After we've got our workstation opened, function init() sets up the program for our current resolution, then changes the mouse pointer to the hand icon.

In order to do this, we first need to get the resolution. We do this with the call:

```
res = Getrez();
```

This returns an integer from *0* to *2*. A value of *0* means the screen is currently in low resolution; a value of *1* in-

dicates medium resolution; and a value of *2* tells you you're in high resolution. This function is defined in the osbind.h file, and is a part of the XBIOS.

In low resolution, the screen dimensions are 320x200. In medium, the horizontal resolution is doubled, giving us a screen 640x200. Finally, in high resolution, both the horizontal and vertical resolutions are doubled (as compared to low res), yielding a screen 640x400. These relationships are important if we're going to write software compatible with all three resolutions.

Let's say we're in low resolution and we draw a rectangle with the coordinates *20 20, 60 20, 60 40, 20 40* and *20 20* (these are the coordinate pairs you would load into the pxy[] array before calling v_pline()). Now we switch to medium resolution and draw the same rectangle.

What happened? The rectangle is only half as long, right? This is because the horizontal resolution has been increased by a factor of 2; the screen pixels are half as wide, so they produce a rectangle half as long. If we want the rectangle the same size in medium resolution as in low (and in the same place on the screen), we have to double the value of the horizontal coordinates. A rectangle drawn in medium resolution between the coordinates *40 20, 120 20, 120 40, 40 40* and *40 20* will look like one drawn with the previous coordinates in low resolution.

Now let's use the medium resolution coordinates to draw the same rectangle in high resolution. Whoops! The figure is the same length, but now it's only half as high. No surprise, right?

I told you earlier the vertical dimension of a high resolution screen was twice that of low or medium, remember? If we want to draw that same rectangle yet again, but in high resolution, we must multiply the vertical coordinates by a factor of 2, giving us *40 40, 120 40, 120 80, 40 80* and *40 40*.

Text output isn't immune to changes in resolution, either. In medium resolution, text is half as wide as in low, while high resolution, which uses a different font, yields text the same width as that in medium resolution, but half as high.

How's all this handled in init()? Well, let's see. Once we get the resolution with a call to Getrez(), we use the returned value in a switch statement to set h_factor (horizontal factor), v_factor (vertical factor) and t_factor (text factor) to their appropriate values. We'll use these values in calculating screen coordinates for the resolution we're in.

Some of the shapes to be drawn by our program have coordinates hardcoded into arrays. This saves us from setting up a pxy array each time we draw one of these shapes; we can, instead, pass the address of the array that contains the coordinates.

To prevent some later calculations, we immediately modify these arrays for our current resolution. The for loop near the bottom of init() accomplishes this, by multiplying each element of the array by one of the factors initialized by the switch statement. The figures whose coordinates are stored in these arrays will then be displayed properly in any resolution.

Of Mice and C.

The function init()'s last task is to change the mouse form from the arrow to the hand. The call that accomplishes this is:

```
graf_mouse (form,mouse_form);
```

Here, form is an integer value from the table below and mouse_form is the address of a 35-element array containing the data for the mouse form. At this point, we're not going to discuss this array, since it pertains to user-defined mouse forms rather than those supplied by the system. We'll discuss custom mouse forms in a future installment of **C-manship**.

The acceptable values for form are as follows:

```
0   Arrow
1   Line cursor
2   Bee
3   Pointing hand
4   Flat hand
5   Thin crosshair
6   Thick crosshair
7   Outlined crosshair
255 User-defined mouse form
256 Hide mouse form
257 Show mouse form
```

Any value from *0* to *7* will yield the mouse form shown. A value of *255* directs the function toward a user-defined

mouse form stored in the mouse_form[] array. A value of *256* removes the mouse form from the screen, and a value of *257* restores it. As we'll see later, the ability to hide the mouse form is critical when drawing on the screen.

The graf_mouse() function is a part of GEM's AES libraries.

Menus and varmints with buttons.

The main program loop, found in do_menu(), utilizes the mouse for menu selection. The outer while loop repeats the menu process until the user wishes to exit the program, while the inner while loop samples the mouse until one of the buttons is pressed.

Also within the inner loop is a call to mouse_print(). This function (found at the end of the listing) prints the coordinates of the mouse in the upper left corner of the screen (actually, it'll print *any* two integers). I use this function to help me find the mouse X,Y-positions I need for my test statements. For instance, when writing this month's sample program, I used mouse_print() to determine what coordinates fell within each of the menu selections. Once the program was completed, I thought that, rather than delete mouse_print() from the listing, I'd leave it for you to fool with. What a guy, huh?

Also, there are a couple of interesting function calls in mouse_print(). One of them, v_gtext(), we'll be using extensively, since it's the VDI function that displays text.The syntax for this call is:

```
v_gtext (handle,x,y,string);
```

The integers *x* and *y* are the location the text is to be printed, and *string* is a pointer to the text (you may use a string literal within the call by enclosing it in quotes). Don't forget that an array name (a string is an array of character) *is* a pointer.

Since v_gtext() will handle only strings, how do we output other forms of data to the screen? What if we're writing a game and need to display a score? No problem. All we have to do is convert the data we want to print into a string. The following example will prepare an integer for printing with v_gtext():

```
sprintf (s,"%d",i);
```

The parameter s is the address of the string where the function is to store the converted data (don't forget to leave space for the null!) The rest of the parameters are the same as for printf(). If you're a little fuzzy on that, reread the first **C-manship**, in issue 39.

Getting back to do_menu(), once a button press is detected, a series of if. . .else statements check which button was pushed and the location of the mouse at the time. The VDI function that returns the mouse status is:

```
vq_mouse (handle,&button,&mx,&my);
```

The parameter handle is, of course, the handle that was returned by the v_opnvwk() call. The parameters &button, &mx and &my are the addresses of integer variables that will hold the button pushed, the mouse's X-position and the mouse's Y-position, respectively. The value returned in button will be *0* if no button is pressed, *1* if the left button is pressed, *2* if the right button is pressed, and *3* if both buttons are pressed.

After we exit the inner while loop, we check for a button value of *1* (left button pressed). If the left button was pressed, we then check the mouse coordinates at the time the button was pressed, to see if the pointer was within one of our menu selections. If it wasn't, repeat retains its true condition, and the outer while loop is repeated. If the mouse pointer was within the menu, we perform the appropriate function, redraw the menu, then return to the main while loop (repeat is still true). If button equals *2* (right button pressed), we set repeat to *0*, which breaks us out of the main loop and returns us to main(), where we close the virtual workstation and then return to the desktop.

Notice that, when checking for mouse coordinates, we're utilizing h_factor and v_factor. The horizontal and vertical mouse coordinates are dependent on the current resolution, just as when drawing a shape. We must multiply each coordinate in the if statements by the appropriate factor.

Text effects.

The ST has several built-in text effects you can use to enhance your programs. Text can be printed bold, light intensity, skewed, underlined, outlined, or any combination of the above. The function do_effects() in the sample program demonstrates these effects.

First, a call to v_hide_c() hides the mouse form, then v_clrwk() clears the screen. The text color is set with the call:

```
vst_color (handle,color);
```

In this, color is an integer from *0* up to the maximum colors available for the current resolution (you know what handle is, right?)

Next, we set the text height (we'll cover this function a little later) and enter the loop that prints the text.The different effects are set with the call:

```
vst_effects (handle,effect);
```

Here, the bits of the integer effect are set as below:

Bit	Value	Effect
0	1	Bold
1	2	Light
2	4	Skewed
3	8	Underlined
4	16	Outlined

Note that the value in the bit column is the number of the bit to set, *not* the value to send to the function. You need to do some binary arithmetic to arrive at the decimal values shown in the second column. Any combination of effects can be used by adding the values together. For instance, if you want just bold text, the parameter effect in the above call should be set to *1*; if you want underlined *and* bold text, effect should be set to *9 (1+8)*; for skewed, outlined, bold text, effect needs the value *21*, and so on.

Text height.

As I mentioned earlier, the ST is capable of displaying text in many different heights. Best of all, you may mix these heights on the screen in any way you wish. To set the height of text to be printed, use the call:

```
vst_height (handle,height,&char_w,
        &char_h,&cell_w,&cell_h);
```

The integer height is the requested height, and the parameters &char_w, &char_h, &cell_w and &cell_h are pointers to integer. Respectively, the values returned in these addresses are: the character width, the character height (from the base line to the top of the cell), the cell width and the cell height. In the sample listing, since we don't need this information, we just return all these values to our old standby, dummy.

Another function we can use to set text height is:

```
vst_point (handle,point,&char_w,
           &char_h,&cell_w,&cell_h);
```

Here, point is the height of text in points (a point equals 1/72 inch). Other parameters are the same as for vst_height().

Text rotation.

The GEM operating system allows text to be printed at any angle. Unfortunately, the ST implementation of GEM allows rotation in 90-degree increments only. To set the base line rotation of the text, use the call:

```
vst_rotation (handle,angle);
```

The integer angle is the angle of rotation in tenths of degrees. Because of the limitation placed on this function for the ST, this value must be *0, 900, 1800* or *2700.*

In the sample listing, the function do_rotate() demonstrates the use of text rotation. Handy for graphs!

Mouse prestidigitation.

In all cases, before we draw something on the screen, we must hide the mouse form. If we don't, we may find a block of the old screen pasted in over the new one as soon as the mouse is moved.

This may seem peculiar at first, but the logic behind it is simple. In order to allow mouse movement, the operating system must save for later redraw the section of the screen covered by the mouse cursor. When the mouse is again moved, the screen is restored by reading back the saved block. The saved screen block remains unchanged if we draw to the screen, so when the mouse is moved and GEM pastes in the old block, we may find a portion of the old screen coming back to haunt us.

The VDI provides the following functions for turning the mouse form on and off:

```
v_hide_c (handle);
v_show_c (handle);
```

There's something to keep in mind when using these functions. Every call to v_hide_c() must have a corresponding call to v_show_c()—unless, of course, you don't plan to see your mouse again. This doesn't mean you can't call v_hide_c() twice in a row; it just means that if you *do* call it twice in a row, you must also call v_show_c() twice to get your mouse back.

Breaktime.

That covers it for this month. Now that you've learned a good deal about the VDI and how to use a mouse, you have the tools to begin some serious GEM programming. The best way to become confident with these tools is to use them. So, until next month, practice what you've learned. //

Listing 1.
C listing.

```
/*****************************************/
/*        C-MANSHIP, ST-LOG #10          */
/*              LISTING 1                */
/*      DEVELOPED WITH MEGAMAX-C         */
/*****************************************/
#include <osbind.h>

#define BLACK 1
#define RED 2
#define GREEN 3
#define HOLLOW 0
#define SOLID 1
#define HAND 3
#define NORMAL 0


int work_in[11], work_out[57];
int contrl[12], intin[128];
int ptsin[128], intout[128], ptsout[128];
int mouse_form[35];

int rec1[] = {106,150,206,50};
int rec2[] = {108,148,204,52};
int line1[] = {108,84,204,84};
int line2[] = {108,116,204,116};

int res, h_factor, v_factor, t_factor;
int handle, dummy;

main()
{
  appl_init();
  open_vwork();
  init();
  do_menu();
  v_clsvwk(handle);
  appl_exit();
}

do_menu()
{
  int repeat, button, mx, my;

  repeat = 1;
  draw_menu();
  while (repeat) {
    button = 0;
    while (button == 0) {
      vq_mouse(handle,&button,&mx,&my);
      mouse_print (mx,my);
    }
    if (button == 1) {
      if (mx)112*h_factor && mx(199*h_factor) {
        if (my)54*v_factor && my(81*v_factor) {
          do_effects();
          draw_menu();
        }
        else if (my)86*v_factor && my(113*v_factor) {
          do_height();
          draw_menu();
        }
        else if (my)118*v_factor && my(145*v_factor) {
          do_rotate();
          draw_menu();
        }
      }
    }
    else if (button == 2)
      repeat = 0;
  }
}

do_effects()
{
  int x, y, effect, b_effect, n_effect, height;

  v_hide_c (handle);
  v_clrwk (handle);
  vst_color (handle,BLACK);
  if (res == 0)
    height = 4;
  else
    height = 8;
  vst_height (handle,height,&dummy,&dummy,&dummy,&dummy);
  b_effect = 1;
  for (x=5*h_factor; x(260*h_factor; x+=62*h_factor) {
    n_effect = 1;
    for (y=25*v_factor; y(126*v_factor; y+=25*v_factor) {
      effect = b_effect | n_effect;
      vst_effects (handle,effect);
      v_gtext (handle,x,y,"EFFECTS");
      n_effect (<= 1;
    }
  b_effect (<= 1;
  }
  v_show_c (handle);
```

$79.95

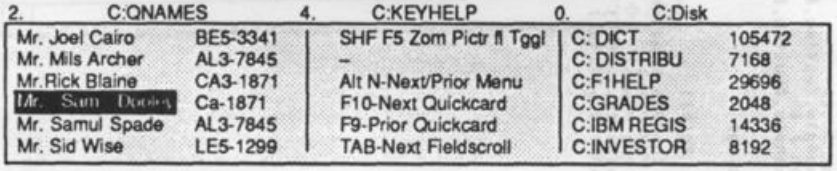
Zoomracks II: an integrated productivity tool thats fun to use.

60 day money back guarantee!

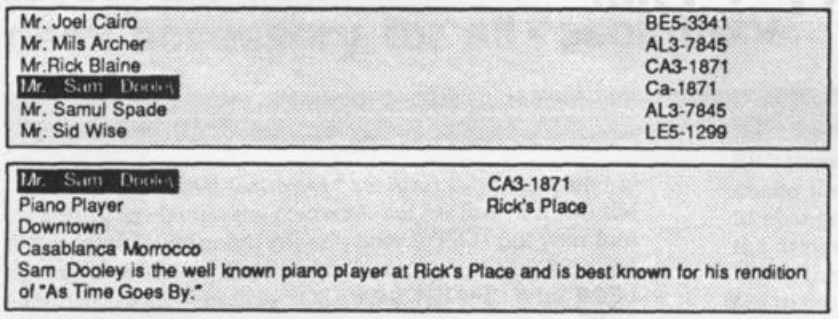
Now with math, report formatting, improved mail merge, Degas file displays, easier to learn.

Zoomracks is an integrated productivity tool based on a simple and familiar concept. You keep your information--names and addresses, invoices, correspondence, or whatever-- in cards in racks like those next to factory time clocks. It is easy to use, helps you structure your problems, and it can handle an exceptionally wide variety of applications.

Zoomracks lets you zoom back to see the first line of several cards, and several racks next to each other--compressed for display with its Smart Zoom feature.



Here you get an overview of three racks. Pressing one key zooms you in on one rack; pressing another, zooms you in on one card. To go someplace on the screen or execute a command, just click the mouse.



What the reviewers said about Zoomracks (version I)

"Zoomracks, a $79.95 wonder...from Quickview Systems"
– **Newsweek**

"The simplest thing in the world to use... very much recommended
–*Jerry Pournelle*, **Byte**

"Truly innovative, ... flexible... extremely easy to use..."
– **InfoWorld**

"practical and even fun ... you can always visualize your data as you want --- from a broad overview of the whole database to a specific detail of a single field..."
– *Art Leyenberger*, **Compute!**

"99% of the database needs in home and small business computers do not require the power and prorgramming features of DBASE III. In fact, by the time you learn DBASE III, you could have all your database needs filled by Zoomracks.

For those of us with two left thumbs...It is very hard to lose data with this well thought out system... less than $100 and worth 10 times that amount"
– *Don Terp*, **ST Business**

You can be an anarchist...and this program will bring order to your chaos...It is useful enough to become a standard equivalent to Lotus 1-2-3 in the industry.
– *Jim Bumpus*, **Atari ACE**

"*****...the more I use Zoomracks, the more I like it, and the more I can see to do with it.... will be with us for a long time, finding a solid niche in both home and office."
– *Richard Keller*, **ST Applications**

Zoomracks has established a productivity standard in design, concept, originality, and functionality."
– *Alan Glick*, **JBUG** (Boston Computer Society)

Zoomracks is totally in a class of its own... I found it so useful I considered buying another ST for my office.
– *John Leon*, **HASTE Electronik Letter**

CIRCLE #112 ON READER SERVICE CARD

Project Organizer
Data Base
Calculator
Word Processor

Power without the programming

POWERFUL: Relational data base, word processor, mail merge, arithmetic, display Degas files, report formatting, organizer, macros, 250 line fields, import-export data.
FLEXIBLE: Modify templates and report formats, copy between fields and racks, view multiple racks, zoom in on a rack or card.
EASY TO USE: No field sizes to specify, few frustrating error messages, visual interface, Doesn't require programming. Few concepts to learn: everything based on cards in racks. You can undo commands
EASY TO LEARN: on-line tutorial, on-line help, 180 page manual, use Commands or menu. Works with or without mouse.
FUN: For people who hate data bases. Point and zoom in on cards, racks. Toggle commands let you try and untry things.

Uses are limited only by your imagintion: Ask some one who uses it.

OFFICE: Mailing lists, Form letters, invoicing, agendas, projects, tickler files, payables, receivables, credit records, correspondence, schedules, appointments, office procedures, check registers, employee records.
EDUCATION: annotated bibliographies, lesson planning, test banks, dissertations, organize notes for class, books, anything you do with index cards.
HOME: Address and phone lists, insurance catalogue, credit card records, shopping lists, Christmas card records, videotape/records catalogue, recipe cards, checkbook.
SALES: prospect files, tickler file, call reports, mailing lists, sales order entry.
Coming soon: Zoomracks Starter Packs with templates and forms for the office, home, school, collectors etc. ($19.95 each).

Available at your local ST retailer or mail this coupon to **Quickview Systems**, Dept. A, 146 Main St., Suite 404 Los Altos, CA 94022
California residents add 6.5% sales tax.

☐ Zoomracks II* $119.95 ($149.95 after 12/25)
☐ Zoomracks I $79.95 (Upgraded)

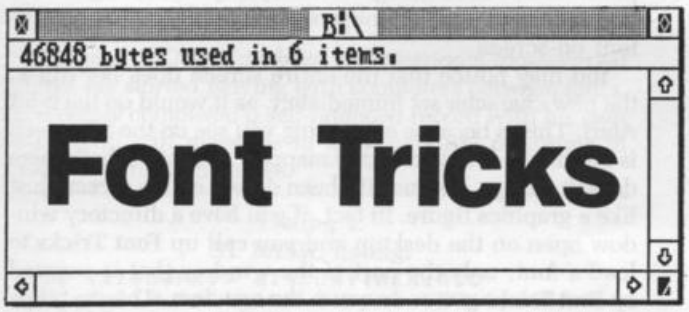
I understand by ordering direct I can return Zoomracks for a full refund within 60 days if not satisfied for any reason.

Name _____
Address _____
City _____
State _____ Zip _____
☐ Check enclosed ☐ Visa ☐ MC
Card # _____ Exp date: _____

*Zoomracks II lets you save macros and create multiple print formats. Registered Zoomracks owners can upgrade to Zoomracks I ($14.95) orZoomracks II ($79.95)

Zoomracks is not copy protected.

```
  button_wait();
}

do_height()
{
  int height, x, y;

  v_hide_c (handle);
  v_clrwk (handle);
  vst_effects (handle,0);
  for (height=1; height<27; ++height) {
    x += 8; y += 7;
    vst_height (handle,height,&dummy,&dummy,&dummy,&dummy);
    v_gtext (handle,x*h_factor,y*v_factor,"Height");
  }
  v_show_c (handle);
  button_wait();
}

do_rotate()
{
  int angle;

  v_hide_c (handle);
  v_clrwk (handle);
  vst_height (handle,8,&dummy,&dummy,&dummy,&dummy);
  for (angle=0; angle<2701; angle+=900) {
    vst_rotation (handle,angle);
    v_gtext (handle,160*h_factor,96*v_factor,"ROTATION");
  }
  vst_rotation (handle,0);
  v_show_c (handle);
  button_wait();
}

draw_menu()
{
  int height;

  v_hide_c (handle);
  v_clrwk(handle);
  draw_rec (rec1,GREEN,SOLID,0);
  draw_rec (rec2,BLACK,HOLLOW,0);
  v_pline (handle,2,line1);
  v_pline (handle,2,line2);
  vst_height (handle,10,&dummy,&dummy,&dummy,&dummy);
  vst_color (handle,RED);
  vst_effects (handle,NORMAL);
  v_gtext (handle,110+152*t_factor,72*v_factor,"EFFECTS");
  v_gtext (handle,116+152*t_factor,104*v_factor,"HEIGHT");
  v_gtext (handle,116+152*t_factor,136*v_factor,"ROTATE");
  v_show_c (handle);
}

draw_rec(rec,fcolr,inter,style)
int rec[];
int fcolr,inter,style;
{
  int rxc[4];
  int x;

  for (x=0; x<4; ++x)
    rxc[x] = rec[x];
  vsf_color(handle,fcolr);
  vsf_interior(handle,inter);
  vsf_style(handle,style);
  v_bar(handle,rxc);
}

open_vwork()
{
  int i;

  handle = graf_handle(&dummy,&dummy,&dummy,&dummy);
  for (i=0; i<10; work_in[i++] = 1);
  work_in[10] = 2;
  v_opnvwk (work_in, &handle, work_out);
}

init()
{
  int x;

  res = Getrez();
  switch (res) {
    case 0 :
      h_factor = 1;
      v_factor = 1;
      t_factor = 0;
      break;
    case 1 :
      h_factor = 2;
      v_factor = 1;
      t_factor = 1;
      break;
    case 2 :

      h_factor = 2;
      v_factor = 2;
      t_factor = 1;
  }
  for (x=0; x<4; ++x)
    if (x == 0 || x == 2) {
      rec1[x] = rec1[x] * h_factor;
      rec2[x] = rec2[x] * h_factor;
      line1[x] = line1[x] * h_factor;
      line2[x] = line2[x] * h_factor;
    }
    else {
      rec1[x] = rec1[x] * v_factor;
      rec2[x] = rec2[x] * v_factor;
      line1[x] = line1[x] * v_factor;
      line2[x] = line2[x] * v_factor;
    }
  graf_mouse (HAND,mouse_form);
}

button_wait()
{
  int button, mx, my;

  button = 0;
  while (button == 0)
    vq_mouse(handle,&button,&mx,&my);
  while (button > 0)
    vq_mouse(handle,&button,&mx,&my);
}

mouse_print(mx,my)
int mx, my;
{
   char tx[5], ty[5];

   vst_height (handle,6,&dummy,&dummy,&dummy,&dummy);
   sprintf(tx,"%d ",mx);
   sprintf(ty,"%d ",my);
   v_gtext(handle,20,30,tx);
   v_gtext(handle,52,30,ty);
}
```



A fool-the-TOS desk accessory for all resolutions.

by Charles F. Johnson

One of the nicest features of an Atari 8-bit computer is its ability to change the character set (or font; I'll use the terms interchangeably). All you have to do is put your font data somewhere in memory (let's say its location is called CHBASE), and then do a *POKE 756,CHBASE*. What could be simpler?

When I started programming my ST, the first thing I wanted to do was to change fonts. Everyone told me it couldn't be done. You see, the ST is actually missing the part of its operating system that lets other fonts be loaded and displayed. This section of the OS is called GDOS, and Atari will eventually release it (or so I'm told) in the form of a disk file which must be placed in an AUTO folder to run at boot-up. GDOS will have all the necessary calls to load multiple fonts into RAM—somewhat like a Macintosh does. (GDOS does a lot more than this, but its other functions are beyond the scope of this article.)

This information sent me to the ST keyboard, grumbling "Can't load fonts, eh?" I hate to be told that something is impossible, and I wasn't willing to wait for GDOS, so. . . after much hair-pulling and tooth-gnashing, **Font Tricks** emerged.

Font Tricks is a GEM desk accessory written in 68000 assembly language. Its purpose is to load any 8-bit or **DEGAS** font file, and make TOS accept it as the default. This means that any program at all (e.g., **1st Word**, ST BASIC, **MicroEMACS**, the desktop) will display text with the font you choose. **Font Tricks** also lets you change back to the ST system font if you so desire. And it works in any resolution, color or monochrome.

To use the program, type in Listing 1. Check your typing with **ST-Check** (see page 84), and run it under ST BASIC; there are no resolution-dependent features. This will create the file FONTTRIX.ACC on drive A. If you want to change the destination drive or the filename, change the assignment to *filename$* in Line 100 (the first line). When the program ends, it will print the message *file written* in the output window. Make sure the destination disk for the file has space for a 3328-byte file.

The ST handles its character sets very differently from the Atari 8-bit models. There are actually three fonts contained in the TOS ROMs: a 6×6 font used for the small text below the icons on the desktop, an 8×8 font for low- and medium-resolution color modes, and an 8×16 font for high-resolution monochrome. Also, the layout of the font data itself is different. On the 8-bit computer, a font is just a series of bytes that defines how each character will look. The first byte in the font corresponds to the top line of the first character, second byte is the second line in the first character, and so on, with 8 bytes making up a single character. The ST font data is laid out so that the first byte is the top line of the first character, but the second byte is the top line of the second character, the third byte is the top line of the third character, etc.

An ST font also has a "header" consisting of 88 bytes, which contains control information about how the font is displayed. This header allows different font sizes, proportional fonts and other futuristic goodies, but the default ST fonts are all "monospaced." Monospaced just means that every character is defined in a block that's the same size, so the letter *i* (for example) is the same width as the letter *M*.

ST fonts also have two tables of offsets, the Character

Offset Table and the Horizontal Offset Table. You can see there's a lot to creating really ST-compatible fonts! If you're interested, there's more info about ST font layouts in the Atari Developer's Kit, or *ST Internals* from Abacus.

How *Font Tricks* does its trick.

I'm going to assume that you have some prior knowledge of ST programming in this explanation—otherwise, this article could quickly take on truly monstrous proportions. Specifically, I'll assume you know how to call the Line A routines. The only one we need to use is $A000, the INITIALIZATION call.

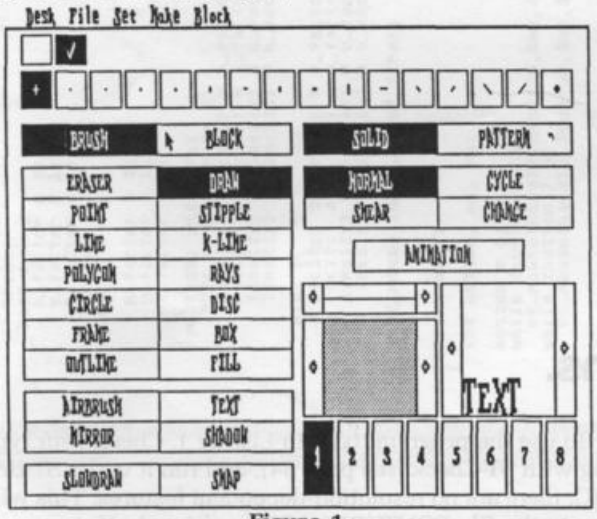


Figure 1.
***DEGAS Elite* screen showing *Font-Tricks*-loaded font.**

One of the longwords in the system font headers is a pointer to the font data. At first, I assumed I could just load a font file from disk, convert it to ST format, and change that pointer to my new font. After the Line A INIT call, register A0 contains a pointer to the address of the Line A system variables, and register A1 contains a pointer to a table with addresses of the three system font headers. I issued the INIT call and got that address, and . . . It turns out that those headers are all in ROM, and not changeable! There went my first idea, down in flames.

So I started using SID (the debugger that comes with the Developer's Kit) to look through the low memory areas of my ST and, lo and behold, I found another set of font headers, this time in RAM (he cackled gleefully). Not only that, there was a pointer to the start of the RAM 8x8 header, just below the start of the Line A variable area. There's also a pointer that determines which font TOS applications will use (non-GEM programs). Note that the pointer GEM uses is a pointer to a font header, while the TOS pointer points to the actual font data. Load a font into RAM, convert it to ST format, change all these pointers—*voilà!*—custom fonts on the ST.

These pointers are somewhat of a mystery to me; I can't find any documentation on them anywhere. I suspect they're hooks for GDOS to use. This method has worked on every ST with TOS in ROM that I've tried, and it should work with future revisions—as long as they don't change the negative displacement of the "mystery" pointers from the Line A variables. If it doesn't work with future versions of TOS. . .why, by then, we'll all have GDOS. The point will be moot. (Right, Atari?) Anyway, for right now, this accessory will do the job if you want to change your font on-screen.

You may notice that the entire screen does not flip to the new character set immediately, as it would on the 8-bit Atari. This is because everything you see on the ST screen is bit mapped, not character mapped. The stuff that's there doesn't change, because it's been drawn on the screen, just like a graphics figure. In fact, if you have a directory window open on the desktop and you call up **Font Tricks** to load a font, only the part of the window that is covered by **Font Tricks** gets redrawn in the new font. The next time you do something that causes the directory windows to be updated, the new font will take over.

Check out the assembly language listing included on this month's disk edition, to see exactly how the font pointers are changed.

One caution about *DEGAS* fonts.

Fonts created for the **DEGAS** drawing program use an 8x16 format, which is double the vertical resolution compared to an 8-bit font. When **Font Tricks** loads a **DEGAS** font, it checks to see which resolution you're in. If you're in high resolution, the 8x16 **DEGAS** format is used. If you're in low resolution, however, every other byte of each character must be thrown away to create an 8x8 font. Properly constructed **DEGAS** fonts have a flag at the end of the file after the font data, which tells whether the font may be scaled to half-height.

If **Font Tricks** determines that a **DEGAS** font can't be scaled properly, you'll see an alert box informing you of this, and the font will not be installed. If you do manage to load a font that looks a bit strange (or even unreadable), call up **Font Tricks** again, even if you can't read the text, and click on the right-hand button. This will restore the system font, and you can try another font.

If a **DEGAS** font looks almost right, you can always use the Font Editor program included with **DEGAS**. A few touch-ups can produce some very pleasing fonts!

How to change your default font.

Font Tricks also has the capability to load a font you specify at boot-up time. This enables you to automatically install any font, **DEGAS** or 8-bit, as the GEM default. To make **Font Tricks** automatically load a font, you must create a text file called FONT.DEF on your boot disk in the main directory. You can use **1st Word** to create this file, as long as you save it with the Word Processor Mode turned off. The first thing in FONT.DEF should be *D* or *8* (for **DEGAS** or 8-bit font types). This should be on a line by itself; in your text editor, just type *D* or *8* followed by RETURN. Then, on the next line, enter the filename of the font you wish to make the default, including drive and pathnames, if you desire. A sample FONT.DEF file might look like this:

```
8
\FONTS\8BTFONTS\COMPUTER.FNT
```

or another example:

```
d
C:\OZARK.FNT
```

When the **Font Tricks** accessory is initialized, it looks in the main directory of the current drive for FONT.DEF. If found, it loads and installs the font specified in the file, before you even hit the desktop!

Charles F. Johnson has been a professional musician all his life. He started playing with computers between tours. He recently completed a self-financed record with his own band—and still manages to work on GEM programs in 68000 assembly language.

Listing 1.
ST BASIC listing.

```
100 filename$="a:\FONTTRIX.ACC"
110 fullw 2:clearw 2:open "R",1,filena
me$,16
120 field#1,16 as bin$:record=0:gotoxy
 0,0
125 print "creating file '"filename$"'
:"
130 readline:
140 hxd$=""
150 for i=1 to 16:read byte$:if byte$=
"*" then goto endit
160 code=val("&H"+byte$):hxd$=hxd$+chr
$(code):next
170 lset bin$=hxd$:record=record+1:put
 1, record
180 print ".";:goto readline
190 endit:
200 close 1:print "file written":end
1000 data 60,1A,00,00,07,72,00,00,04,7
6,00,00,21,22,00,00
1010 data 00,00,00,00,00,00,00,00,00,0
0,00,00,2E,7C,00,00
1020 data 2D,06,4E,B9,00,00,00,0C,42,B
9,00,00,26,F2,42,B9
1030 data 00,00,26,F6,42,B9,00,00,26,F
A,42,B9,00,00,26,FE
1040 data 23,FC,00,00,08,78,00,00,08,E
6,61,00,07,32,33,F9
1050 data 00,00,25,E4,00,00,0C,04,23,F
C,00,00,08,82,00,00
1060 data 08,E6,33,F9,00,00,0C,04,00,0
0,24,E4,23,FC,00,00
1070 data 07,72,00,00,27,02,61,00,07,0
6,33,F9,00,00,25,E4
1080 data 00,00,0C,08,3F,3C,00,04,4E,4
E,54,8F,33,C0,00,00
1090 data 0C,0A,3F,3C,00,19,4E,41,54,8
F,D0,3C,00,41,13,C0
1100 data 00,00,0B,67,13,C0,00,00,0B,A
7,0C,79,00,02,00,00
1110 data 0C,0A,67,00,00,10,32,3C,00,0
8,2A,7C,00,00,0A,5A
1120 data 60,00,00,0C,32,3C,00,10,2A,7
C,00,00,0A,62,30,3C
1130 data 00,08,7A,03,28,7C,00,00,0A,7
A,38,DD,51,CD,FF,FC
1140 data 2A,7C,00,00,0A,82,7A,08,61,0
0,04,26,A0,00,23,E8
1150 data FF,EA,00,00,0B,EC,2A,68,FE,3
C,23,ED,00,4C,00,00
1160 data 0B,F0,2A,6D,00,54,23,ED,00,4
C,00,00,0B,F4,0C,79
1170 data 00,02,00,00,0C,0A,67,00,00,1
0,2A,69,00,04,2A,3C
1180 data 00,00,01,FF,60,00,00,0C,2A,6
9,00,08,2A,3C,00,00
1190 data 03,FF,2A,6D,00,4C,28,7C,00,0
0,14,CC,28,DD,51,CD
1200 data FF,FC,23,FC,00,00,08,8C,00,0
0,08,E6,42,B9,00,00
1210 data 24,E4,61,00,06,2A,2A,7C,00,0
0,A0,00,28,7C,00,00
1220 data 08,49,7A,0C,10,15,B0,14,67,0
0,00,06,52,8D,60,F4
1230 data 20,4D,B9,0D,66,E6,51,CD,FF,F
A,20,08,23,C8,00,00
1240 data 0C,00,22,08,C2,BC,FF,FF,FF,F
E,20,41,55,88,22,10
1250 data B0,81,66,F8,23,C8,00,00,0B,F
C,2A,7C,00,00,0B,5E
1260 data 42,85,61,00,03,30,6B,00,00,4
E,33,C0,00,00,0C,06
1270 data 2A,7C,00,00,0C,BC,7A,46,61,0
0,03,2A,61,00,03,40
1280 data 2A,7C,00,00,0C,BF,0C,39,00,4
4,00,00,0C,BC,67,00
1290 data 00,22,0C,39,00,64,00,00,0C,B
C,67,00,00,16,0C,39
1300 data 00,38,00,00,0C,BC,66,00,00,0
E,61,00,01,EC,60,00
1310 data 00,06,61,00,01,10,23,FC,00,0
0,08,96,00,00,08,E6
1320 data 23,FC,00,00,0C,2C,00,00,27,0
2,61,00,05,72,0C,79
1330 data 00,28,00,00,0C,2C,66,DE,30,3
9,00,00,0C,34,B0,79
1340 data 00,00,0C,08,66,D0,3F,3C,00,1
9,4E,41,54,8F,33,C0
1350 data 00,00,0C,1A,3F,3C,00,00,2F,3
C,00,00,0C,7C,3F,3C
1360 data 00,47,4E,41,50,8F,23,FC,00,0
0,0A,6A,00,00,0B,E8
1370 data 61,00,03,86,42,79,00,00,0C,1
0,61,00,04,C2,0C,79
1380 data 00,07,00,00,0C,0E,66,00,00,0
A,61,00,00,62,60,00
1390 data 00,26,0C,79,00,08,00,00,0C,0
E,66,00,00,0A,61,00
1400 data 01,22,60,00,00,12,0C,79,00,0
9,00,00,0C,0E,66,00
1410 data 00,06,61,00,02,1C,33,FC,00,0
3,00,00,0C,0C,61,00
1420 data 04,06,3F,39,00,00,0C,1A,3F,3
C,00,0E,4E,41,58,8F
1430 data 2F,3C,00,00,0C,7C,3F,3C,00,3
B,4E,41,5C,8F,28,79
1440 data 00,00,0B,FC,28,B9,00,00,0C,0
0,60,00,FF,2A,28,79
1450 data 00,00,0B,FC,28,BC,00,00,08,5
7,20,7C,00,00,0B,67
1460 data 61,00,02,5C,4A,39,00,00,0C,1
C,66,00,00,04,4E,75
1470 data 0C,79,00,01,00,00,25,E6,67,0
0,00,04,4E,75,2A,7C
1480 data 00,00,0C,1C,42,85,61,00,01,C
C,6B,00,00,94,33,C0
1490 data 00,00,0C,06,2A,7C,00,00,0C,B
C,2A,3C,00,00,08,02
1500 data 61,00,01,C2,6B,00,00,7A,61,0
0,01,D4,28,7C,00,00
1510 data 0E,BC,2A,7C,00,00,14,EC,7A,5
F,0C,79,00,02,00,00
1520 data 0C,0A,67,00,00,40,20,7C,00,0
0,0C,BC,4A,68,08,00
1530 data 66,00,00,14,2A,7C,00,00,07,8
1,3A,3C,00,01,61,00
1540 data 02,5E,60,00,00,3C,78,07,1A,9
4,54,8C,DB,FC,00,00
1550 data 01,00,51,CC,FF,F4,9B,FC,00,0
0,07,FF,51,CD,FF,E8
1560 data 60,00,00,1A,78,0F,1A,9C,DB,F
C,00,00,01,00,51,CC
1570 data FF,F6,9B,FC,00,00,0F,FF,51,C
D,FF,EA,61,00,00,DE
1580 data 4E,75,28,79,00,00,0B,FC,28,B
C,00,00,08,67,20,7C
1590 data 00,00,0B,A7,61,00,01,88,4A,3
9,00,00,0C,1C,66,00
1600 data 00,04,4E,75,0C,79,00,01,00,0
0,25,E6,67,00,00,04
```

```
1610 data 4E,75,2A,7C,00,00,0C,1C,42,8
5,61,00,00,F8,6B,C0
1620 data 33,C0,00,00,0C,06,2A,7C,00,0
0,0D,BC,2A,3C,00,00
1630 data 02,00,61,00,00,F0,6B,A8,2A,7
C,00,00,0C,BC,2A,3C
1640 data 00,00,01,00,61,00,00,DE,6B,9
6,2A,7C,00,00,0F,BC
1650 data 2A,3C,00,00,01,00,61,00,00,C
C,6B,84,61,00,00,E0
1660 data 28,7C,00,00,0D,BC,2A,7C,00,0
0,14,EC,7A,5F,0C,79
1670 data 00,02,00,00,0C,0A,67,00,00,1
E,78,07,1A,9C,DB,FC
1680 data 00,00,01,00,51,CC,FF,F6,9B,F
C,00,00,07,FF,51,CD
1690 data FF,EA,60,00,00,22,78,07,1A,9
4,DB,FC,00,00,01,00
1700 data 1A,9C,DB,FC,00,00,01,00,51,C
C,FF,EE,9B,FC,00,00
1710 data 0F,FF,51,CD,FF,E2,61,00,00,0
4,4E,75,A0,00,21,7C
1720 data 00,00,14,CC,FF,EA,2A,68,FE,3
C,0C,79,00,02,00,00
1730 data 0C,0A,67,00,00,0E,2B,7C,00,0
0,14,CC,00,4C,60,00
1740 data 00,0E,2A,6D,00,54,2B,7C,00,0
0,14,CC,00,4C,4E,75
1750 data A0,00,21,79,00,00,0B,EC,FF,E
A,2A,68,FE,3C,2B,79
1760 data 00,00,0B,F0,00,4C,2A,6D,00,5
4,2B,79,00,00,0B,F4
1770 data 00,4C,4E,75,3F,05,2F,0D,3F,3
C,00,3D,4E,41,50,8F
1780 data 4A,40,4E,75,2F,0D,2F,05,3F,3
9,00,00,0C,06,3F,3C
1790 data 00,3F,4E,41,DF,FC,00,00,00,0
C,4A,80,4E,75,3F,39
1800 data 00,00,0C,06,3F,3C,00,3E,4E,4
1,58,8F,4A,40,4E,75
1810 data 78,01,DB,FC,00,00,00,10,36,1
5,C6,C0,3A,C3,36,15
1820 data C6,C1,3A,C3,51,CC,FF,F2,51,C
D,FF,E6,4E,75,24,7C
1830 data 00,00,0C,1C,72,0F,42,5A,51,C
9,FF,FC,4E,75,23,C8
1840 data 00,00,0B,F8,61,E8,23,FC,00,0
0,08,A0,00,00,08,E6
1850 data 23,F9,00,00,0B,F8,00,00,27,0
2,23,FC,00,00,0C,1C
1860 data 00,00,27,06,61,00,02,18,42,8
0,20,79,00,00,0B,F8
1870 data 10,10,90,3C,00,41,3F,00,3F,3
C,00,0E,4E,41,58,8F
1880 data 20,79,00,00,0B,F8,54,88,22,7
C,00,00,0C,3C,7A,3F
1890 data 4A,10,67,00,00,08,12,D8,51,C
D,FF,F6,7A,3F,0C,21
1900 data 00,5C,67,00,00,06,51,CD,FF,F
6,52,89,42,11,48,79
1910 data 00,00,0C,3C,3F,3C,00,3B,4E,4
1,5C,8F,4E,75,23,CD
1920 data 00,00,27,02,33,C5,00,00,24,E
4,23,FC,00,00,08,AA
1930 data 00,00,08,E6,60,00,01,A8,23,F
C,00,00,08,B4,00,00
1940 data 08,E6,23,F9,00,00,0B,E8,00,0
0,27,02,61,00,01,90
1950 data 33,F9,00,00,25,E6,00,00,0C,1
2,33,F9,00,00,25,E8
1960 data 00,00,0C,14,33,F9,00,00,25,E
A,00,00,0C,16,33,F9
1970 data 00,00,25,EC,00,00,0C,18,42,7
9,00,00,0C,0C,61,00
1980 data 00,86,33,FC,00,01,00,00,0C,0
C,61,00,00,7A,23,FC
1990 data 00,00,08,D2,00,00,08,E6,42,7
9,00,00,24,E4,33,FC
2000 data 00,01,00,00,24,E6,33,F9,00,0
0,0C,12,00,00,24,E8
2010 data 33,F9,00,00,0C,14,00,00,24,E
A,33,F9,00,00,0C,16
2020 data 00,00,24,EC,33,F9,00,00,0C,1
8,00,00,24,EE,23,F9
2030 data 00,00,0B,E8,00,00,27,02,61,0
0,01,04,23,FC,00,00
2040 data 08,BE,00,00,08,E6,42,79,00,0
0,24,E4,23,F9,00,00
2050 data 0B,E8,00,00,27,02,61,00,00,E
6,33,F9,00,00,25,E4
2060 data 00,00,0C,0E,4E,75,23,FC,00,0
0,08,C8,00,00,08,E6
2070 data 33,F9,00,00,0C,0C,00,00,24,E
4,33,FC,00,10,00,00
2080 data 24,E6,33,FC,00,02,00,00,24,E
8,33,FC,00,40,00,00
2090 data 24,EA,0C,79,00,02,00,00,0C,0
A,67,00,00,0E,33,FC
2100 data 00,08,00,00,24,EC,60,00,00,0
A,33,FC,00,10,00,00
2110 data 24,EC,33,F9,00,00,0C,12,00,0
0,24,EE,33,F9,00,00
2120 data 0C,14,00,00,24,F0,33,F9,00,0
0,0C,16,00,00,24,F2
2130 data 33,F9,00,00,0C,18,00,00,24,F
4,60,00,00,62,23,FC
2140 data 00,00,08,DC,00,00,08,E6,33,F
9,00,00,0C,0E,00,00
2150 data 24,E4,42,79,00,00,24,E6,33,F
9,00,00,0C,12,00,00
2160 data 24,E8,33,F9,00,00,0C,14,00,0
0,24,EA,33,F9,00,00
2170 data 0C,16,00,00,24,EC,33,F9,00,0
0,0C,18,00,00,24,EE
2180 data 33,F9,00,00,0C,10,00,00,24,F
0,42,79,00,00,24,F2
2190 data 23,F9,00,00,0B,E8,00,00,27,0
2,60,00,00,02,22,3C
2200 data 00,00,08,E6,20,3C,00,00,00,C
8,4E,42,4E,75,20,20
2210 data 46,6F,6E,74,20,54,72,69,63,6
B,73,21,00,5B,33,5D
2220 data 5B,20,54,68,69,73,20,66,6F,6
E,74,20,63,61,6E,6E
2230 data 6F,74,20,62,65,20,73,63,61,6
C,65,64,20,7C,20,74
2240 data 6F,20,68,61,6C,66,2D,73,69,7
A,65,3B,20,61,20,6D
2250 data 6F,6E,6F,63,68,72,6F,6D,65,2
0,7C,20,6D,6F,6E,69
2260 data 74,6F,72,20,69,73,20,6E,65,6
5,64,65,64,2E,20,53
2270 data 6F,72,72,79,21,20,20,7C,20,5
D,5B,20,54,68,61,74
2280 data 27,73,20,4C,69,66,65,21,20,5
D,00,20,46,4F,4E,54
2290 data 20,54,52,49,43,4B,53,21,20,0
0,50,72,65,73,65,6E
2300 data 74,73,2E,2E,2E,00,62,79,20,4
3,68,61,72,6C,65,73
2310 data 20,46,2E,20,4A,6F,68,6E,73,6
F,6E,00,49,6E,73,74
2320 data 61,6C,6C,20,77,68,69,63,68,2
0,66,6F,6E,74,3F,00
2330 data 44,45,47,41,53,00,38,2D,42,4
9,54,00,53,59,53,54
2340 data 45,4D,00,00,00,49,54,45,4D,2
0,53,45,4C,45,43,54
2350 data 4F,52,00,4C,4F,41,44,20,44,4
5,47,41,53,20,46,4F
2360 data 4E,54,00,4C,4F,41,44,20,38,2
D,42,49,54,20,46,4F
2370 data 4E,54,00,00,00,0A,00,00,00,0
1,00,00,00,00,00,23
2380 data 00,01,00,01,00,01,00,00,00,1
8,00,02,00,01,00,00
```

```
2390 data 00,00,00,17,00,00,00,01,00,0
1,00,00,00,5A,00,00
2400 data 00,02,00,02,00,00,00,34,00,0
1,00,01,00,01,00,00
2410 data 00,36,00,00,00,05,00,01,00,0
0,00,32,00,01,00,02
2420 data 00,01,00,00,00,33,00,09,00,0
1,00,01,00,00,00,2A
2430 data 00,06,00,01,00,01,00,00,00,2
F,00,08,00,01,00,01
2440 data 00,00,00,00,24,CC,00,00,26,E
4,00,00,24,E4,00,00
2450 data 25,E4,00,00,27,02,00,00,28,0
2,00,00,00,00,00,00
2460 data 00,00,00,00,00,00,1F,E7,C0,8
3,FC,00,3F,F7,E1,C7
2470 data FE,00,60,77,F1,CC,0E,00,C0,7
7,79,D8,0E,00,E0,77
2480 data 3D,DC,0E,00,FF,77,9F,DF,EE,0
0,FF,F7,CF,DF,FE,00
2490 data FF,77,E7,DF,EE,00,F8,77,E3,D
F,0E,00,FC,77,E1,DF
2500 data 8E,00,FC,57,E1,5F,8A,00,78,2
3,C0,8F,04,00,00,00
2510 data 00,00,00,00,00,00,00,00,00,0
0,00,01,C7,22,F2,00
2520 data 00,02,28,B6,8A,00,00,02,08,A
A,8A,00,00,02,08,A2
2530 data F2,00,00,02,28,A2,82,00,00,0
1,C7,22,81,00,00,00
2540 data 00,00,00,00,00,00,00,00,00,0
0,C0,03,FF,1F,F8,00
2550 data E0,07,FF,BF,FC,00,E0,07,FF,B
F,FC,00,E0,07,03,B8
2560 data 00,00,E0,07,03,B8,00,00,F0,0
7,83,BC,F8,00,F8,07
2570 data C3,BE,7C,00,FC,07,E3,BF,1C,0
0,FE,07,F3,BF,9C,00
2580 data FF,E7,FF,BF,FC,00,FF,D7,FE,B
F,F4,00,7F,E3,FF,1F
2590 data F8,00,00,00,00,00,00,00,00,0
0,00,00,00,00,2F,A8
2600 data 4E,00,00,00,22,2C,51,00,00,0
0,22,2A,50,00,00,00
2610 data 22,29,53,00,00,00,22,28,D1,0
0,00,00,C2,28,4F,00
2620 data 00,00,00,00,08,FE,00,06,00,1
6,00,00,00,00,00,01
2630 data 00,00,09,82,00,06,00,16,00,0
0,00,00,00,01,00,00
2640 data 07,FE,00,00,08,47,00,00,08,4
7,00,05,00,06,00,02
2650 data 11,80,00,00,FF,FF,00,0C,00,0
1,00,00,08,0A,00,00
2660 data 08,47,00,00,08,47,00,05,00,0
6,00,02,11,80,00,00
2670 data FF,FF,00,16,00,01,00,60,00,1
6,00,E8,00,50,00,60
2680 data 00,26,00,E8,00,A0,FF,FF,00,0
1,00,09,00,14,00,00
2690 data 00,10,00,02,11,00,00,00,00,0
0,00,00,00,00,00,02
2700 data FF,FF,FF,FF,00,17,00,00,00,0
0,00,00,0A,06,00,01
2710 data 00,01,00,05,00,01,00,03,FF,F
F,FF,FF,00,17,00,00
2720 data 00,00,00,00,0A,14,00,06,00,0
1,00,05,00,01,00,04
2730 data FF,FF,FF,FF,00,15,00,00,00,0
0,00,00,0A,22,00,0C
2740 data 00,01,00,09,00,01,00,05,FF,F
F,FF,FF,00,1C,00,00
2750 data 00,01,00,00,07,EF,00,0C,00,0
2,00,0E,00,01,00,06
2760 data FF,FF,FF,FF,00,15,00,00,00,0
0,00,00,0A,3E,00,0C
2770 data 00,03,00,10,00,01,00,07,FF,F
F,FF,FF,00,1C,00,00
2780 data 00,00,00,00,08,20,00,05,00,0
5,00,13,00,01,00,08
2790 data FF,FF,FF,FF,00,1A,00,05,00,0
0,00,00,08,34,00,02
2800 data 00,07,00,07,00,02,00,09,FF,F
F,FF,FF,00,1A,00,05
2810 data 00,00,00,00,08,3A,00,0B,00,0
7,00,07,00,02,00,00
2820 data FF,FF,FF,FF,00,1A,00,25,00,0
0,00,00,08,40,00,14
2830 data 00,07,00,07,00,02,00,00,0A,6
A,46,4F,4E,54,2E,44
2840 data 45,46,00,41,3A,5C,2A,2E,46,4
E,54,00,00,00,00,00
2850 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2860 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2870 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2880 data 00,00,00,41,3A,5C,2A,2E,46,4
E,38,00,00,00,00,00
2890 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2900 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2910 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2920 data 00,00,00,00,00,00,00,02,06,0
6,06,06,06,06,04,0A
2930 data 04,06,04,06,04,06,04,0A,04,0
E,12,06,08,0E,0E,0C
2940 data 0C,10,0C,0C,08,26,0C,04,06,1
0,20,18,06,10,06,10
2950 data 08,0C,0C,16,04,06,04,0C,08,0
6,10,0A,0E,04,0A,0C
2960 data 14,14,10,0A,0E,0E,06,0A,06,0
6,0A,0E,0C,10,06,18
2970 data 06,0A,0A,0E,4E,06,06,0A,0E,0
C,0E,06,12,12,16,06
2980 data 0A,4E,0E,0A,10,0C,0C,0C,1E,1
6,30,10,08,04,06,04
2990 data 06,04,0C,16,08,26,10,06,06,0
4,0A,04,06,04,0A,04
3000 data 06,04,06,04,06,04,06,0C,0A,0
4,06,08,06,04,06,04
3010 data 06,04,06,04,06,04,0A,04,06,0
6,04,0A,04,08,04,06
3020 data 04,08,08,08,08,0C,0C,06,04,0
6,04,06,04,06,04,0A
3030 data 04,06,04,06,06,04,06,04,06,0
4,06,04,06,04,06,06
3040 data 04,0A,01,84,04,04,04,04,04,0
1,0E,0E,0E,04,04,14
3050 data 04,04,48,18,18,18,18,18,18,1
8,18,0C,00,1A,1A,1A
3060 data 1A,1A,1A,1A,1A,1A,1A,1A,1A,1
A,1A,1A,1A,1A,1A,1A
3070 data 1A,1A,1A,1A,1A,1A,1A,1A,1A,1
A,1A,1A,1A,1A,1A,1A
3080 data *
```

●

ST CHECKSUM DATA.

(see page 84)

```
 100 data  228, 341, 226, 535, 653, 14
5, 960, 264, 372, 466, 4190
 190 data  221, 513, 631, 585, 900, 95
2, 853, 751, 881, 766, 7053
 1080 data  815, 908, 753, 744, 738, 4
4, 842, 80, 950, 698, 6572
 1180 data  763, 36, 995, 817, 863, 92
, 127, 999, 766, 820, 6278
```

```
 1280 data  859, 762, 800, 768, 777, 8
35, 892, 803, 893, 782, 8171
 1380 data  679, 704, 683, 736, 824, 9
03, 905, 843, 765, 817, 7859
 1480 data  853, 779, 823, 905, 808, 7
35, 932, 265, 949, 300, 7349
 1580 data  912, 781, 757, 928, 838, 9
26, 867, 833, 964, 894, 8700
 1680 data  114, 989, 162, 75, 6, 819,
 880, 117, 881, 914, 4957
 1780 data  860, 971, 870, 990, 357, 6
4, 957, 872, 764, 848, 7553
 1880 data  892, 36, 980, 949, 882, 86
2, 860, 909, 869, 775, 8014
 1980 data  869, 918, 791, 838, 908, 7
44, 888, 881, 868, 828, 8533
 2080 data  895, 848, 757, 907, 819, 8
54, 817, 838, 899, 860, 8494
 2180 data  837, 775, 819, 891, 920, 8
71, 915, 16, 885, 876, 7805
 2280 data  839, 820, 881, 932, 922, 7
90, 793, 825, 841, 614, 8257
 2380 data  553, 581, 551, 567, 580, 5
80, 777, 645, 33, 130, 4997
 2480 data  408, 422, 973, 676, 835, 7
98, 785, 312, 3, 257, 5469
 2580 data  559, 704, 665, 769, 650, 5
91, 702, 791, 638, 861, 6930
 2680 data  872, 557, 926, 913, 583, 9
47, 934, 687, 972, 935, 8326
 2780 data  587, 949, 945, 612, 955, 7
61, 775, 548, 549, 550, 7231
 2880 data  742, 552, 546, 547, 607, 7
05, 697, 734, 740, 768, 6638
 2980 data  769, 684, 631, 628, 654, 6
06, 674, 721, 847, 848, 7062
 3080 data  199, 199
```

•

Zoomracks

QUICK VIEW SYSTEMS
146 Main Street
Los Altos, CA 94022
(415) 965-0327
ST $79.95

by Matthew J.W. Ratcliff

Zoomracks is a database designed for fast, easy access to large amounts of information. The program is based on the "card-rack" concept. Each record of information is thought of as a card, and a file of cards is referred to as a rack. Each card can have multiple fields, providing quick references to your data. The author thought the concept so unique that **Zoomracks** is patented.

Before continuing here, I think it best to define the new terms **Zoomracks** uses in a simple, straightforward way (something overlooked in the manual). A *ZoomRack* is a data file. A *QuickCard* is a record of information in a data file. A *FieldScroll* is simply a variable length field. When you do a *SmartZoom*, you're moving from a macro (wide-angle) to a micro view (close-up) of a record, or vice versa. This is a unique feature.

Zoomracks has an unfamiliar user interface. It does not take advantage of the GEM windowing operating system; commands are accessed via menus and submenus. In many cases, there are several different ways to get the job done. While this adds flexibility, it makes learning the system confusing. I found the on-line tutorial helpful, but the manual is poorly organized. Nowhere in the text are all commands listed in a concise manner. I also found the text confusing and redundant.

For example, both chapters 3 and 6 tell you how to load and run the program. The text wastes your time explaining the obvious: "When you start up **Zoomracks**, you are telling your system that you want to start the program **Zoomracks**." Before long I gave up on the manual and worked through the tutorial, which gives a flavor for the program, its speed and the power of macros. The demo can be made continuous, by selecting one of the macros in the tutorial session. I found the manual's chapter summaries the most useful, and dug into the text only *when all else failed*.

As you learn the program, you may turn on the auto-help feature with the F4 key. Anytime a command is selected, a 4-line "index card" is displayed at the bottom left of the screen, giving a brief summary of the command's features. You'll have to invest a lot of time getting comfortable with this program, but will then find it fast and powerful.

The most frequent command sequence I used while learning was ESCAPE-L: "escape from what I'm doing to a long-shot view of everything." It's easy to get lost among the menus and submenus (which appear as a single-key command prompt line at the bottom of the display), and ESCAPE-L is an elegant way to get back to safety — without botching up.

When you fire up **Zoomracks**—without all the extra demos and autoloading macros provided on the disk—you're presented with two "racks." Rack zero is always the disk directory rack. All filenames with a .ZRX extension appear here. Rack one is reserved for your macro definitions.

As many as nine racks may be open and displayed at once. Each appears as a column of text, growing narrower with every new rack shown. When you look at many racks at a time, each one's allotted display space becomes too narrow to show a complete field. The SmartZoom abbreviates these fields by stripping out vowels, letting you see lots of information from different racks at the same time—and still read most of it. When specific information is needed, just zoom in.

The F1 key toggles between single- and multiple-rack display. When in single-rack mode, the currently active ZoomRack fills the screen. The F2 key toggles between single- and multiple-card display modes. In the multi-card mode, just the first FieldScroll of each QuickCard is seen. In single-card mode, the currently highlighted QuickCard is completely shown. Every field has a name, to be turned on or off with the F9 key.

The card you're currently viewing may be edited at any time. You may also add a new QuickCard of information to the current rack, or change rack field names and arrangement. Your rack is automatically sorted on the first field, alphabetically. If you have a ZoomRack of addresses, you may re-sort on the zip-code field and print labels for presorted mail.

If you also keep phone numbers and comments in your address rack, you must copy your ZoomRack, then paste it into an empty rack. Once copied, you're prompted to rename it. You must go to the ALT-M (modify menu) and press F5 (format rack menu). From there, you'll select ALT-A (arrange rack menu). At this point, you may use ARROW keys to select a field, and the DELETE key to remove unwanted FieldScrolls, before printing your labels. A bit involved, to say the least.

If there's any item of interest, it should be in a separate, named field—otherwise, sorts and major updates will be a *major*

pain. You may search the first line of the current FieldScroll of all your QuickCards, but if that FieldScroll has more than one line, the extra lines won't be searched.

Zoomracks is fast, since all its work is done in RAM. When you quit the program, you may save or discard all changes. When you save, any racks that were updated in the last edit session will be saved to disk. Free RAM is indicated by a "gas gauge" at the bottom of your screen.

There's a full screen editor that's quite easy to use. I found it handy for creating a quick letter, but still prefer my word processor, **1st Word**. Unlike **Zoomracks**, with **1st Word** you don't need to know a single function, control, or alternate keypress to access all the features. A program need not be complicated to use to be powerful.

I don't agree with the **Zoomracks** manual, that the program is appropriate for payroll records, sales order entry, credit card budget records, or any other application handling numerical data. **Zoomracks** has no arithmetic capabilities, and won't meet your needs in such tasks.

I think it's ideal for mailing lists, outlines, inventories— anything requiring a lot of information handling which can be committed to a "form." **Zoomracks** lets you create forms with FieldScroll definitions that are easy to fill in. For others less familiar with computers, the program can be made more user friendly via proper automatic macro file controls.

The manual isn't very useful beyond its summaries. You'll do better with the online tutorial, working along with the automatic help enabled until you're familiar with the features.

Overall, I found **Zoomracks** fast, bug-free, and fairly easy to use—once I had the basic commands down pat.

Zoomracks Version II came in just before presstime. It still doesn't use the familiar GEM windowing system interface, but you can select any menu option via mouse control. The menu area at the bottom of the screen has been expanded by several lines, so all options from the current menu are displayed. It's easy to navigate this program with the mouse control, learning the keyboard controls more gradually. Math features have been put into this version, also you have two "registers" to work with, and may perform addition, subtraction, multiplication and division on them. A running total can be carried through fields on a card, or multiple cards. These updates, among others we haven't had time to explore, vastly improve on the user interface (while retaining keyboard controls for the expert's fast navigation of the program, and the new math features make it more appropriate for some of the business applications mentioned earlier. A very helpful function key template and command reference card are provided, as well. Next issue a review of **Zoomracks II**. //

ST Golf Games

Hole-In-One Golf

ARTWORX SOFTWARE CO., INC.
150 N. Main Street
Fairport, NY 14450
(800) 828-6573
$29.95

Leader Board

ACCESS SOFTWARE INC.
2561 S. 1561 West
Woods Cross, UT 84087
(801) 298-9077
$39.95

Mean 18

ACCOLADE
20833 Stevens Creek Blvd.
Cupertino, CA 95014
(408) 446-5757

by Rick Teverbaugh

Fore!. . .or perhaps more accurately. . . Three!

When it's time to put away the bag, the clubs and the shoes, ST owners no longer need also closet the enjoyment they get from the links. Now there are three golf games that, to varying degrees, make excellent use of the power and graphics capabilities of the ST.

In order of increasing price, the games are: **Hole-In-One Golf**, **Leader Board** and **Mean 18**.

Leader Board, first of all, is a bit too easy. When I booted the game, I picked the most difficult setting, played all the four included courses end-to-end in a 72-hole marathon sitting, and ended up with a 268, or 20 under the 288 par.

With **Leader Board**, if you hit into the water, the ball is moved back to the same spot via a penalty stroke. In sand or in the rough, you blast your way out; trees you hit around, under or over—all without losing stroke or distance.

Another **Leader Board** weakness comes from one of its strengths. Its graphics are the most outstanding of the three programs reviewed here. Situated slightly behind the golf figure, your view is almost the same as being right in his shoes. There is, however, no utility to create your own holes, or to alter in any way the courses included, presumably because the graphics are so complex. **Leader Board** is the only one of the three without this feature.

The game has three difficulty levels. At the easiest, you can't hook or slice the ball, and the wind won't affect the shot. The next level introduces hooks and slices, and the top level throws everything at the golfer.

If you're having non-golfing friends over and want to wow them with the beauty of one of your game programs, just reach for **Leader Board**.

Hole-In-One has a couple of glaring problems. The program doesn't feel much like golf, partly due to its overhead point of view. You're never given the opportunity to get your feet wet or get sand in your shoes.

The ball bounces unrealistically. On one hole, a driver produced a hop that cleared the green on the first bounce. It's also a slow and painful process (at least in terms of strokes) to blast out of the trees. It's always better to take the drop ball option—since the program doesn't have an axe among the clubs in your bag.

The greens have no contour. You just line up the putt and gauge the strength. Finally, the 18-hole course included is a nightmare for any true golf fan. There's a par-three hole that you can't birdie unless you can chip a shot in from off the green. Many of the holes look good aesthetically, but don't stand up under repeated playings.

Once the program loads, you can select the hole you'd like to begin with. (On the other two games, you must start at the first tee.) You'll see the entire hole and all the options in a single screen. The mechanics of setting up a shot are handled at the bottom of the screen, using a joystick.

Club selection is difficult, because there is no figure to indicate the number of yards to the hole. Even a reasonable guess won't be much help, because in the manual there's no chart of approximate distances for each club.

It's possible to create your own course. You can set the par, hole number and complete layout of the hole. There's only one type of rough, but there are three types of trees and an easy fill routine to make design a snap.

Even with the problems mentioned, the **Hole-In-One** program is versatile and challenging—a good buy.

Mean 18 is the Mercedes of golf simulations. I call it a simulation instead of a game, because it *feels* like golf. The only complaint I have is that the action of the ball, both bouncing on the fairway and rolling on the green, leaves a little to be desired.

Three famous courses come with **Mean 18**: Scotland's St. Andrews, Augusta and Pebble Beach. Of the three, Pebble Beach is my favorite, since the greens are so much tougher to putt.

Mean 18 is the only one of the three to provide true sloping of the greens. Not only is it possible to have almost an unlimited number of breaks on the green, it's also possible to have a putt that must allow for two or three such breaks. In other words, the challenge doesn't stop once the ball reaches the green. Putting is the only part of the game that takes the overhead perspective, and it's necessary in order to see all the breaks on the green.

There are four levels of difficulty. Beginner and expert settings determine whether the caddy suggests the right club and the difficulty of hitting the ball straight. There are also regular and championship tees.

Stroke, match, or best ball competition can be played for up to four golfers. The program keeps track of scoring in each of the three modes. A nice feature is the Hall of Fame display, which keeps a permanent record of the top ten scores, the player's initials, the mode selected, which tees were used and the date of the game. It's also possible to print out the Hall of Fame or the individual score cards.

All the above features help make **Mean 18** top-notch, but the Course Architect is one of the program's strongest points.

It's possible to custom design each tree, set any type of slope on the green and even paint the background that will used on each course. I named one course after my wife, and I could even inscribe her nickname, Marf, on the clubhouse roof.

I'm sure that, at some point, someone (maybe even one of these three companies) will surpass **Mean 18** in realism and versatility. But if it gets much more realistic, we'll have to mail in greens fees and take lessons from a pro to compete. //

Rick Teverbaugh has a B.A. in journalism, eleven years' experience as a sports writer and four as a computer games reviewer. He is past Editor of Computerland's newsletter and former Midwest Editor for Electronic Games *magazine.*

The Animator

by Keith Enge
MICHTRON
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700
$39.95

by D.F. Scott

It's not easy to give a MichTron product a less-than-average review. Here's a company that rushed to publish good-quality products (**M-Disk, MichTron BBS**) back when the promised sea of ST software barely resembled a farm pond. Here's a company willing to take risks—especially when it published exclusively for the small but loyal group of Sanyo 555 owners. So I'm not going to madly fire away at **The Animator**, like Rambo during his last visit to a small town, but...

The Animator resembles a product of the era before the Atari 400/800s, before Jay Miner's player-missile graphics, back when programmers struggled to trick the 6502 CPU into performing a simple block screen memory transfer—without having it collapse in a pile of random bits. Using the ST (a product of advanced design and for greater possibilities) shouldn't be a struggle, even for a hobbyist.

Reading the manual is perhaps the foremost difficulty in using the program. It takes us on an uneventful romp through kindergarten, just so we can move blocks of memory across the screen. Only through practice and research can you figure out what the *programs* do (**The Animator** is run in three stages). Even if you already own the disk, here's what you're expected to do...

Using **Neo-Chrome v0.5**, you compose the frames belonging to the object to be animated. Then you must draw a single-pixel-wide border, in color 15 (the rightmost color on **Neo-Chrome**'s color bar, usually white) around each frame, to define a size the interpreter can deal with. Each border may be as tall as you wish, but its interior width is limited to a multiple of 16 pixels.

You then break each frame into three equal-area components, which the manual chooses to call "masks." The task is time-consuming.

Once you've defined your frames and masks, you save them as a **Neo-Chrome** picture. A minor inconvenience, you'll have to translate the file extension .NEO to .PIC for the sake of the interpreter, which believes the latter is **Neo**'s current file extension.

Exactly what frame is shown, when and for how long, is defined with ANIMATE2.PRG. You state where the initial frame is to be shown on the screen, how much the next one is to move, and so on, by defining a set of step-by-step instructions. You can define a portion of these steps as a "subroutine" or a "loop" for the convenience of branching and repetition. Since this program considers one unit of "delta X" as 16 pixels (one "column"), any movement to the right or left is one giant leap for your animated object. Therefore, if you wanted your single-column object to move gracefully, one pixel-at-a-time, you'd have to compose at least fifteen two-column-wide frames depicting the object in each state of pixel-wide movement.

Finally, after you've created your animation sequence and saved it using the extension .AN, you can use ANIMATE3.PRG to set the speed of frame-redraw. Now sit back and watch your completed animation. You can set it against a **Neo-Chrome** background (with the extension .PIC), but it can't move—only your object can. The question remains: why go through all that trouble just for a "sprite?"

I feel MichTron is to be commended for its effort in distributing ST software. But now that there's a bunch of it out, MichTron should slow down and concentrate on quality, utility and integrity. //

D.F. Scott is an artist, writer, educator and programmer living in Oklahoma City. He is currently engaged in the study of quantum physics, computing, and other ways in which elementary particles interact with each other. Otherwise, he fills infinite pieces of paper.

Graphic Artist

PROGRESSIVE COMPUTER APPLICATIONS, INC.
2002 McAuliffe Drive
Rockville, MD 20851
(301) 340-8398
Requires disk drive and printer $199.95
Font Editor $79.95
Font Paks $49.95

by Arthur Leyenberger

When the Atari ST was first shown some two years ago, my immediate reaction was that the 68000-microprocessor-based computer would revolutionize graphics. I specifically had Computer Aided Design (CAD) applications in mind. The amount of memory the computer could address, the speed of graphic display, plus the high-resolution monochrome monitor, all contribute to what I thought would be the beginnings of a CAD system workstation. Of course, all that remained was for some software publisher to create a sophisticated CAD program for the ST.

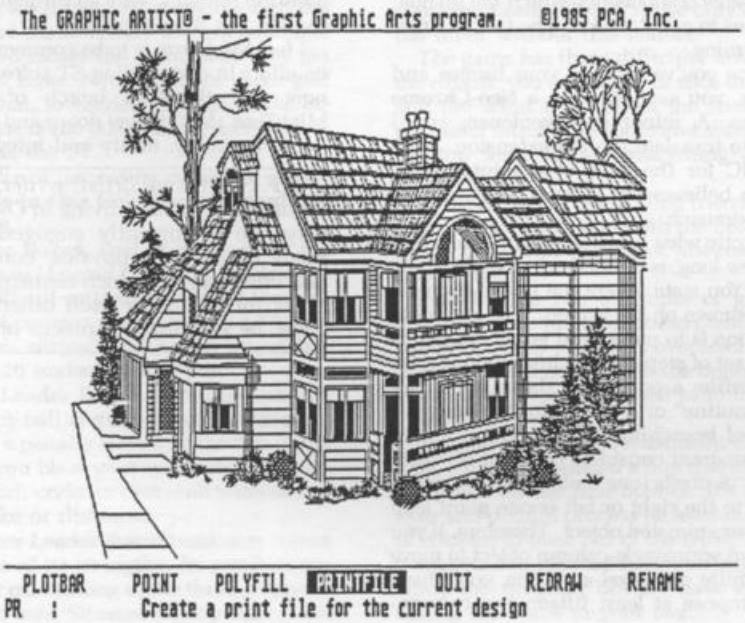
Progressive Computer Applications, Inc. (PCA) has done just that. The **Graphic Artist** is a CAD system for the ST, providing the user with power, versatility and value. It combines the features of CAD with those of desktop publishing, in a single integrated environment. Unlike some so-called integrated software, which appear to have various pieces "glued" together under one program name, **Graphic Artist** features are available interchangeably within one document.

As long as you have an ST, one disk drive and a dot-matrix printer, you can immediately take advantage of the **Graphic Artist** to enter the world of desktop publishing and computer aided design. The program is one of the first to use the full potential of a dot-matrix printer—for very impressive results. If you have access to a plotter or laser printer, you'll be even more amazed at what you can do with it.

There are basically three parts to the **Graphic Artist:** CAD, word processor and spreadsheet. The main purpose, creating text and graphics, is achieved primarily from the graphic screen, rather than from the spreadsheet. Similar to **Easy Draw** (Migraph) the CAD environment here uses "elements" or objects. Each element is a separately created entity, which can be stacked or overlaid without destroying the hidden parts.

In contrast, a painting program (such as **DEGAS**) uses a bit-mapped or pixel-oriented approach. Here, drawing on top of another graphic shape obliterates the existing one. Once a new shape is drawn, the previous one cannot be restored.

The **Graphic Artist** allows up to 256 drawing "layers." The layers can be thought of as clear sheets, each held in perfect registration with the others. When used with an RGB monitor in medium resolution, the program permits fifteen colors, (plus one for background), three of which are visible on the background at any one time. These can be used to identify the many layers. Since the final output isn't limited by screen resolution, what you see on-screen is a rough approximation of what will come out on your printer or plotter.



The bottom of the screen contains a subset of the sixty-one **Graphic Artist** commands, from which you can scroll left or right with the arrow keys to see additional commands. In order to execute one of the functions, you can either find the command and press RETURN, or type up to three unique letters of the command. For example, typing *P* places the cursor over the PAN command. Then, typing *AS* would place the cursor over the PASTE command. When more than one command starts with the same beginning letter string, the cursor moves to the first it sees.

When I began using the **Graphic Artist**, I was disappointed that it didn't use traditional GEM drop-down menus. However, as I used the program and my drawings became more complex, I realized the drop-downs would only get in the way and distract me from the primary task: completing my work. PCA's choice of an appropriate interface instead of menus is to be commended.

Help is generally available and given within context. Asking for help at the command level displays several dialog boxes with information on how to execute any command. However, if you have already issued, say, the LINE command, pressing HELP will yield information relative to the command—such as how to enter points and coordinates. Personally, I prefer more specific contextual help, like being given the available options or range of values to enter. It is the casual or novice user who will typically forget these details and seek help.

The UNDO key is used to terminate any command before execution, regardless of how far into the command you've gotten. Unfortunately, once a command is issued, there's no UNDO function to negate the effects of the last command. Any mistakes must be fixed by cutting and pasting, a potentially laborious procedure.

The first step in the creative process of **Graphic Artist** is to define your work area or "world." Once coordinates are entered, you can use the variable spacing grid and snap feature, to help align graphic shapes. The program provides several graphic "primitives" (built-in shapes) for you to use. These consist of points, lines, circles, arcs, pie slices and filled patterns.

Although different line styles can be used, there's no provision for different line thicknesses. Likewise, about ten patterns are provided, but they aren't the GEM patterns we're familiar with—nor is there an option to create a user-defined pattern. These seem like obvious oversights which can be corrected in the next major release of the program.

The **Graphic Artist** distinguishes between symbols and drawings. Symbols are named pictures or text that can be saved separately in a symbol file. They may be scaled or rotated, and may be combined with other symbols. Drawings are collections of symbols, text and graphic primitives saved as a single entity. Various elements of the drawing can be scaled and rotated, to achieve whatever your design requires.

A number of features facilitate the use of the **Graphic Artist**. The zooming function, in particular, is superb. The zoom ratio is continuously variable, via either the mouse or keyboard entry. This gives incredible flexibility in creating and viewing your work. Further, you can fine tune your output with scale and zoom commands, since the exact printer output is determined by the view on-screen.

The spreadsheet portion of the program is hidden from the user unless specifically requested. That's not to say it's unused, though. All commands and inputs made while creating your work are stored in the spreadsheet. By manipulating the rows and columns of the spreadsheet directly, you can alter the on-screen graphics. Although small in comparison to Lotus 1-2-3, **VIP Professional** and the like, the 500-by-500 cell spreadsheet of **Graphic Artist** is unique.

It gives **Graphic Artist** tremendous capability for graphic modeling. Editing the contents of spreadsheet cells by entering data, altering formulas, or using the built-in functions provides an alternative method for creating output. Also, you can display part of the spreadsheet simultaneously with the drawing, by specifying the number of columns you want.

An interesting feature of the **Graphic Artist** is its ability to create presentation graphics. Pie and bar charts are derived from spreadsheet data. What makes it so versatile is that, once a chart is made, it can become a symbol, to be saved in the symbol library and used wherever you want it.

Several text processing functions are provided, for use with your CAD, presentation graphics or desktop publishing work. You can insert a single line of formatted or unformatted text into your drawing. You can choose from a variety of fonts. Once text is entered, it can be moved or copied, since it's contained in the spreadsheet cells. In addition, text can be rotated, justified and printed—as bold, italics or underline characters.

Printing is performed by first saving your work to a print file, then running a separate printing program. The time required to both create the print file and print it varies with the complexity of your drawing. The **Graphic Artist** supports Epson-compatible printers but also contains a driver editor for other dot-matrix printers, Hewlett-Packard plotters and laser printers.

The **Graphic Artist** can also create and execute macro files. A macro file is a set of frequently used commands, executed simply by giving a command and the filename. The extensive demos provided use macro files to demonstrate the various aspects of the program.

The documentation consists of a several-hundred-page, IBM-style looseleaf binder with slipcase. Writing quality is high; indexes are provided for the main manual, as well as for the **Font Editor**. Appendices abound and information can easily be found. Although plenty of examples are sprinkled throughout, the book is not a tutorial. The novice would be well advised to find introductory books on computer aided design and desktop publishing, if additional information on these topics is required.

Font Editor is a separate program, sold separately by PCA. It lets you modify any of the supplied program fonts, or create your own custom fonts. A font is defined slightly differently than in the context of **Graphic Artist**. Since size and pitch (number of characters per inch) is dependent on the overall size your characters will be, the **Font Editor** is used solely to create the style or look of characters in the font.

Ten fonts are supplied with the program, so you may never need to create your own. However, if you do need custom fonts—or are just curious—once you start creating, you may get hooked. The process of font design is straightforward and even fun. Being able to see the character come to life encourages this creativity.

In addition to the **Graphic Artist**'s ten fonts and the ability to design your own via the **Font Editor**, PCA also sells **Font Paks**, containing additional typeset-style fonts for use with the program. The first available set of fonts, **Font Pak 1**, contains four attractive font styles: Helvetica Medium, Nouvelle, Cornate and Roman Times-2. These are characterized by better quality curves in letters like *B* and *R*. Also, lines have thickness—as a letter becomes larger, it gets thicker.

When the **Graphic Artist** was first released, it retailed for $495.00. That may seem high, but not when compared to CAD packages for the IBM PC. Now, the price is $199.95—a remarkable value.

Not only has PCA lowered the price, but they've also substantially improved the program. A number of bugs have been eliminated and many program features improved, such as a faster display of text with graphics, the use of all available memory for text and spreadsheet storage, a reported tenfold decrease in printing time, saving various settings with a file, improved WORLD functioning, the ability to abort printing, and better handling of user mistakes.

For all this, the **Graphic Artist** now requires three disks (unprotected, by the way) to hold the over 1 megabyte of program, font, demo and other files. In addition, the latest version (V1.51) of the program now requires TOS on ROM. The addition of a hard disk, although not mandatory, is highly recommended for speed of file access and convenience.

There are a number of features, in addition to those mentioned above, **Graphic Artist** needs to achieve "superstar" software status. Additional graphic primitives would be handy, like rounded-corner rectangles, ellipses and polygons. An UNDO command for fill is an absolute necessity. It would also be quite useful to have a status command, which would present information about current settings such as WORLD, GRID, ZOOM, SNAP and filename.

The **Graphic Artist** is a highly sophisticated ST program that rivals similar CAD software available for the IBM PC and its compatibles. The ability to painlessly manipulate text, use an endless variety of premade and user-defined fonts, integrate text and graphics within a document, create presentation graphics (pie charts and bar plots)—all these make it an outstanding program. This capability has not previously been available to the home user. //

COMPUTER PALACE WE KNOW ATARI ST!

Have a very MERRY CHRISTMAS

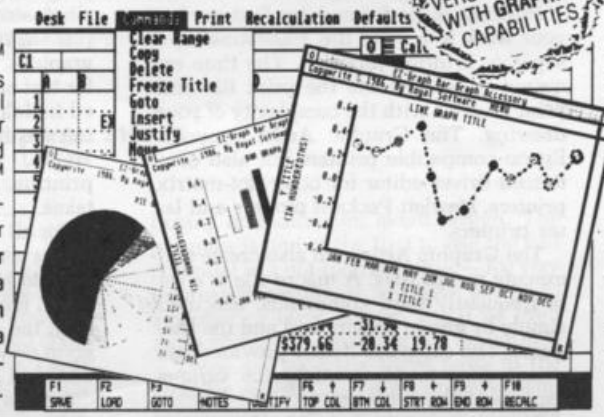
EZ Calc™

PUT EZ CALC™ TO WORK FOR YOU AND BENEFIT FROM THE RESULTS

New!!! Enhanced VERSION 1.2 NOW WITH GRAPHING CAPABILITIES

ONLY $69.95

EZ CALC™ is a fully implemented GEM™ based spreadsheet for home and business use. This is by far the most powerful spreadsheet available for the price. Better yet, all commands are mouse controlled for speed and ease of use. EZ CALC™ also uses **less memory** than other spreadsheets for the ST, leaving more room for your data and formulas. **If you've never used a spreadsheet before, you'll be amazed how easy EZ CALC™ is to learn and use. The experienced user will love the speed of a mouse controlled spreadsheet.** ★ **DEMO AVAILABLE FOR $5.**



FEATURES

- 300 columns by 999 rows
- Extensive use of GEM™ windows
- All commands are under mouse control
- Built in 10 keypad calculator
- On-line help windows (No commands to memorize)
- Built in sort routine
- Developed exclusively for the Atari ST
- 10 macros controlled by the function keys
- Split-screen capabilities
- Note Pad

GEM is a Trademark of DIGITAL RESEARCH, INC.
EZ CALC is a Trademark of ROYAL SOFTWARE.

Help Calc — Only 24.95

- 11 preprogrammed templates for use with **EZ CALC™** or **VIP Professional™**
- **load-and-go** and these templates will take the work out of tedious spreadsheet setup.

Templates include:

- Check Register
- Depreciation schedules
- Investment Portfolio Analysis
- Name & Address directory
- Home Inventory
- Loan Amortization Schedules
- Personal Finance Statement
- and more

VIP PROFESSIONAL is a Trademark of VIP TECHNOLOGIES.

EZ SPELL — Only $39.95 — NEW

For the Atari ST

- Over 130,000 word dictionary, with over 1 million words multilingual.
- Real-time spell checker and stand alone program.
- Postfix and prefix spell checking.
- User dictionary limited only by disk space.

EZ SPELL is a real-time GEM based spelling checker that works with any GEM based word processor, data base, language, etc. Use the stand alone program to check any non-GEM based file. Misspelled word are highlighted within sentences for easy recognition, and can be added to user dictionary. The multi-window display eases error correction by allowing the user to scroll through the text, while the possible correct words and options are continually displayed.

HELPMATE ONLY $29.95

Four desk accessories: Calculator, Notepad, Telephone/Address book and Alarm Clock. Takes up less memory then most accessories. Will work with any GEM based program. Of special interest, the Alarm clock will flash a window up in any GEM based program to remind you of an appointment or whatever. Helpmate is always ready within GEM to help with organizing your desk.

EZ THESAURUS.....Only $39.95 NEW

This real-time or stand-alone program is a window based Thesaurus. Helping the user find just that word that will spruce up an otherwise tedious document. The utility gives the user a list, from a selection of thousands of words, that mean the samething as you have highlighted. If you've just drawn a blank and can't think of a word, let EZ Thesaurus help you find it.

INVENTORY MASTER — NEW

INVENTORY MASTER is an easy to use Gem based inventory program that will do everything from keeping track of inventory to generating detailed printed reports. With GEM pull down menus you can execute fast and easy searches, sorts and printouts. It is so easy you can do it almost without looking at the manual. Additionally the program is written to be easily integrated with Invoice Master, giving you a powerful automatic inventory control system.

INVOICE MASTER — Only $99.95 — COMING SOON

- Create your own invoices
- Use in combination with Inventory Master or by itself
- Print your own invoices on any printer
- Stores more than 200 records per SS/DD disk

Create Your Own Invoices

Invoice Master allows you to print out custom invoices to your printer then subtract the items purchased from inventory through Inventory Master. It is a full GEM implementation, with menus, windows and mouse control. Which makes it much easier to use. You will be amazed at the power and ease with which invoices can be made up with Invoice Master.

PAYROLL MASTER — Only $99.95

- Stores up to 400 employee records per disk.
- State tax information for multiple states
- Prints Employee Checks - Address Labels - W-2 Forms - Employee List - End of Quarter/Year Reports - PLUS MUCH MORE!
- Ram resident for lightning fast access.

Payroll Master is a GEM based payroll program that will give the average user the power to perform any payroll task large or small. All the information is available at a click of a button, so it is fast and easy to operate. No need to buy extra modules because the program has everything you need for completing tasks from printing checks to detailed reports.

THE SCHEDULE — Only $29.95

An Employee Weekly Work Scheduler

- Print a weekly schedule
- Easy to use GEM based program
- Employees names, hours, and dates

COMING SOON

Schedule your employees and then print a weekly schedule showing — employee names, working hours, and dates. See at a glance who will be working. It is GEM based for easy access to weekly schedule information through drop down menus, windows, and mouse control.

EZ GRAPH — ONLY $69.95 — COMING SOON

A professional graphics and statistical analysis package. Graph everything from your finances through stock market quotes. The program will operate within EZ CALC spreadsheet program or as a stand alone professional graphing program.

GREAT BATTLES, 1789-1865 — ONLY $34.95 ..... NEW!!

Relive some of the greatest battles in history, such as Waterloo, Austerlitz, Shiloh and Gettysburg. You make leadership decisions just as General Grant and Emperor Napoleon did. Great sounds of battle and colorfully detailed maps aid in leading you to the final victory.

COMPUTER PALACE

OPEN M-F, 9-6 Sat. 10-4 (Pacific Time)
710 McKinley, Eugene, Oregon 97402

MasterCard VISA AMERICAN EXPRESS

USE YOUR CREDIT CARD & CALL
Toll Free 1-800-452-8013
★ ORDERS ONLY, PLEASE ★

There's never a penalty for using your credit card!
For Information, Call (503) 683-5361
Prices subject to change without notice.

SHIPPING INFO: Minimum $2.90 Ground. $4.75 Air. Actual Cost depends on weight. Call (503) 683-5361 for information.
WARRANTY INFO: Everything that we sell is warrantied by the manufacturer. If any item purchased from us fails to perform properly when you receive it, call us at (503) 683-5361 so that we can assist you. No returned merchandise accepted without authorization. Defective software will be replaced with another copy of the same program, otherwise, no software is returnable.

• 2 Day Air Shipping **AVAILABLE** •

CIRCLE #114 ON READER SERVICE CARD

Z-Time

TERRIFIC PERIPHERALS
129 Sherman Street
Cambridge, MA 02140
(617) 876-2505
$49.95

by Charles Bachand

"You can't possibly have a battery operated clock in that ST," said the suspicious user to his friend. "There isn't any kind of card plugged into the cartridge port!" Unbelieving though he was, he knew his buddy would never say, "How much ya wanna bet?" unless he had *some* ace up his sleeve. And, of course, he did.

Casual ST users may not care whether files are date and time stamped. If they do, they will probably use the control panel desk accessory (if it's been installed on their boot disk), and set the date and time with it. And that's fine—for the casual user.

But for some of us, knowing when a file was created can be of the utmost importance. Software developers keep track of their revisions by the date and time stamp, and hard disk users would go crazy without some sort of chronological indicator. Owners of both the color and monochrome monitors are in even more trouble. When they swap monitor cables, the system reboots and, of course, the date and time must be reset. The addition of a hardware clock is the only real solution.

Z-Time, from Terrific Peripherals, has got to be one of the smallest real-time clocks available for the Atari ST computers. It's so small (shades of Johnny Carson) that it actually fits under one of the ROM chips. The **Z-Time** unit resembles a fat IC socket with a small printed circuit board, CMOS clock chip and a 10-year life span Lithium battery buried in it.

To install **Z-Time**, you simply take your machine apart (I've done it enough times to be able to use the word *simply* in this context), remove the top ROM of the six that you'll find located on the left side of the board, then plug the **Z-Time** unit into the ROM's place, and, last, mount that displaced ROM chip on top of the **Z-Time** unit. *Last* is a bad word to use here, because your final task is to put the computer back together.

And now, folks, it's time to install the clock and see how well it performs. (No, I haven't installed it yet!) If this is the last paragraph in this review—well, you'll all know that something went wrong. Otherwise, I'll be back in a few minutes...

Yeah, I'm back. Did you miss me? The operation was a gigantic success, and the patient is doing very well indeed—heart ticking away and not missing even one beat. The only difficulty that I encountered was in putting the metal shield back on the printed circuit board. The left-hand side of the shield cannot be properly tied down once the clock is in place, but this won't interfere with the operation of the computer. And once the plastic cover is in place, the shield won't even be visible.

When I started writing this review, the **Z-Time** clock could not be installed in a 1040ST, because of the height constraints inside that machine. This problem has been fixed with the installation of a small accessory cable that can also be obtained from Terrific Peripherals.

On a personal note, I would like to say that **Z-Time** is great for left-handed users. These people (myself included) like to manipulate the mouse with their left hand and would constantly be bumping into anything at all that's plugged into the cartridge port—like other companies' clock cards.

Z-Time comes with software to set the clock and calendar after installation, and to adjust the time when daylight savings rolls around. The software to read the clock on power-up is also included and can easily be copied into an AUTO folder for automatic execution. A desktop accessory is also provided, to make the date a somewhat permanent fixture (it can be turned off) in the top right-hand corner of your desktop display. ///

Atari ST 3D Graphics Programming: Concepts and Techniques

by Uwe Braun
A Data Becker Book
ABACUS SOFTWARE
P.O. Box 7219
Grand Rapids, MI 49510
351 pages $24.95

Putting Descartes before the mouse

by Douglas Weir

Data Becker marches on. The latest addition to their series of progamming manuals for the ST, published by Abacus Software, is a how-to handbook on the basics and not-so-basics of 3-D graphics. Those of you who saw the impressive demo programs distributed by Abacus some months ago as an appetizer will have been eagerly awaiting this book. On the whole, it was worth waiting for.

I should begin by telling you that the programs in ***3D Graphics*** are written and presented in 68000 assembly language *only*. This is the only way to achieve routines with an acceptable level of speed. If you don't know assembly language and are not planning to learn it, you will have little use for this book.

3D Graphics is divided into five sections. The first is a short introduction. Section 2 is a quick, practical introduction to the mathematical basis of typical graphics operations. Transformation of Cartesian to other coordinate systems, scaling, rotation around one or more axes, clipping, projection, perspective transformation, and hidden lines and surfaces are covered.

To learn graphics programming from this book, you must thoroughly understand the material in Section 2 before going any further. This will be difficult if you use this book alone. Abacus has made dramatic improvements in proofreading and editing since the first volumes in their ST series. But an unusual degree of verbal precision is required to teach the details of new mathematical techniques, especially when the theory is skipped in favor of immediate practical use. This level has not yet quite been reached by Abacus. I could make no sense out of the introduction to matrix operations beginning on page 19,

where the terms *matrix* and *array* seem to be used interchangeably, and where the syntax is just muddy enough to prevent one from getting a clear view of any of the points discussed.

So I read pages 125-154 of *Microcomputer Graphics* by Roy E. Myers (other equally good manuals are mentioned in the back of the Abacus book, but they're much more expensive). It turns out that matrices and three-dimensional coordinate systems are not so fearsome after all. And once I understood the basics, I found the rest of the Abacus book very easy to understand.

Section 3 gets you started on the mechanics of actually doing graphics on the ST. A line-drawing algorithm is discussed and implemented. Tables are used to implement the sine/cosine function, and, they are fully and clearly explained. These foundations are used to develop the first program in the book, which draws random lines on the ST screen.

Section 4 treats the more advanced topics. Separate programs illustrate rotation around one or two axes, shading, clipping, hidden line removal, and so on. Some new concepts are added from time to time, but these are clearly explained. A graphics driver file contains the basic routines used by the other programs in this section, which must be linked to it. As I mentioned, the source code is AS68 compatible. Abacus devotes a couple of pages to explaining the ins and outs of using the DRI assembler and linker—they seem never to tire of this topic.

For an extra $14.95 (plus $2.00 postage) you can get from Abacus a disk containing all the source code in the book, plus batch files for the assembler and linker. Both the disk versions and the listings in the book will run in medium or high resolution.

Section 5 consists of a few general suggestions for building on the programs presented, to create your own graphics applications. There are appendices covering number systems, matrix multiplication, and so on (these are all well written), a short bibliography of related books, and the skeleton of an index.

There are more than 125 pages of assembly source code in this book. The programs are clearly printed, well commented, planned in a sensible modular fashion, and contain many invaluable assembly-language "tips and tricks." And they work. ST programmers are fortunate to have this book. There are very few substantial graphics manuals available offering practical instruction (i.e., source code on the machine level) for owners of specific computers. A few years ago *Byte* magazine published a 3-D graphics driver for IBM machines, but if you're willing to wade through 8086 code to learn graphics programming, then I want you around the next time I have to move my books and records.

Atari ST Graphics & Sound

by J. Walkowiak
A Data Becker Book
ABACUS SOFTWARE
P.O. Box 7219
Grand Rapids, MI 49510
255 pages $19.95

by Clayton Walnum

Abacus Software certainly deserves a round of applause for their unflagging support. Beginning with their first title, *Presenting the Atari ST*, they've published no less than a dozen books designed to slake every ST owner's thirst for information. Although the books vary in quality, from barely usable to indispensable, no other single publisher has so thoroughly covered the ST. And it'll be a long time, if ever, before someone manages to catch up.

Whether ***Graphics & Sound*** falls into the barely usable category, or is higher on the scale, depends upon what you hope—or think—you're going to get for your twenty bucks.

What *I* thought I was going to get, based on the title, was far from the reality. I expected examples of raster operations; an in-depth study of the VDI functions; perhaps a comprehensive explanation of the screen RAM, including both the single-plane monochrome and the two- or four-plane color incarnations. I wanted to read about screen flipping, character set modification and color register manipulation. Maybe even some special effects with interrupts.

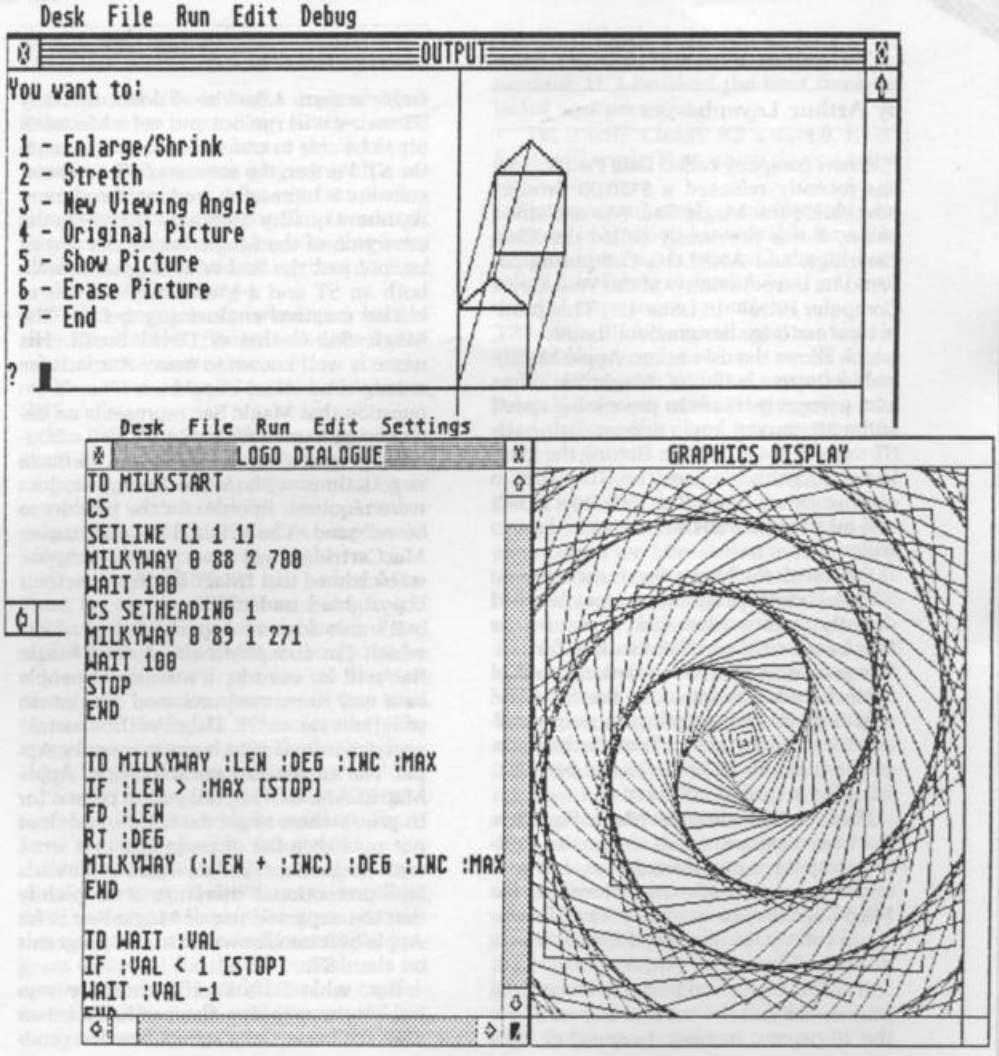
Not quite. The graphics part of ***Graphics & Sound*** is really a primer in general graphics theory. Sure, all the program examples are ST specific, but the information offered can, in most cases, be applied to *any* computer. There are only about ten pages dedicated to GEM, with other minor references scattered throughout. Once again, whether the above is good or bad depends on what you're looking for.

The book explains how to calculate a line between two points, how to manipulate two-dimensional figures—moving them, stretching them, rotating them and mirroring them. There's a good discussion of pie charts and bar graphs, as well as one of the plotting of functions.

Three-dimensional graphics are also covered, including a discussion on perspective, hidden line removal and the plotting of 3-D functions. The reader is even supplied with an interesting mini-CAD program (in BASIC). Stereo graphics are taken up as an adjunct to the 3-D topic, and stereoscopy (the method used with a View Master) and the drawing of figures to be viewed with 3-D glasses (those things always make me feel drugged) are both touched upon. The graphics section is closed with some words about fractals, including sample programs in BASIC, Modula-2 and C. Most of the other program examples, however, are written in the dreaded ST BASIC.

The section on sound can be daunting if you're not on intimate terms with your ST's hardware. Most of the discussion is on a machine language level, with examples in C. The short example programs illustrate manipulation of the sound envelope and present the reader with two (could they spare it?) sound effects. Programmers who prefer BASIC or Logo will find a mere two pages each (not counting listings). This is not the book to buy if you want to learn about sound on your ST, anyway; forty pages is the extent of that coverage here.

In all fairness, this is a decent enough book if you're aware of what you're getting—and you don't expect to write the next **Sundog** after reading it. Is it worth the rather hefty price tag? I don't think so. At $12.95, it might be tempting, but there's not enough information tucked between its covers to warrant spending $19.95. If you're interested in the book, I suggest you browse through it before plunking down your green stuff. ///



ST user

Magic Sac: Is this sorcery for Good or Evil?

by Arthur Leyenberger

A new company called Data Pacific, Inc. has recently released a $130.00 product now called the **Magic Sac**. (As explained below, it was previously called the **MacCartridge**, and **ANALOG Computing** covered its introduction—at the West Coast Computer Faire—in issue 45.) This product is a cartridge designed for the Atari ST, which allows the user to run Apple Macintosh software on the ST. **Magic Sac** offers a 20-percent increase in processing speed and a 30-percent larger screen, using the ST monochrome monitor. Getting the **Magic Sac** to work requires the ST owner to obtain a pair of Macintosh 64K Boot ROMs and insert them into the sockets in the cartridge.

Even with the **Magic Sac** inserted in an ST, Macintosh programs cannot be read directly from a Mac disk, because the Macintosh uses a different disk format. The programs must be copied via a serial cable from a Macintosh to the ST—and only unprotected programs on unprotected disks can be used. Transferring programs takes about ten minutes per disk, with the supplied software.

Data Pacific claims that **Magic Sac** offers the Macintosh owner, as above, the ability to run Mac programs faster and on a larger screen. That's true. However, the **Magic Sac** may be used by some ST owners to run pirated copies of Mac software on their STs.

It seems highly unlikely that an existing Mac owner will run out and buy an ST for the 20-percent increase in speed or for a larger screen. Likewise, I doubt that any ST owner will run out and get a Mac simply to be able to transfer Mac programs to the ST. Further, the amount of ST computer software is increasing, and more and more represent quality programs. So what's the attraction of the **Magic Sac** to the rest of us, not just the few who happen to own both an ST *and* a Macintosh?

The creative engineering behind the **Magic Sac** is that of David Small. His name is well known to many Atarians, as a long-time Atari supporter. There's no question that **Magic Sac** represents an engineering *tour de force*, but it's also a troubling product. From the outset, delicate negotiations with Apple Computer, Inc. were required, in order for the product to be released. The original product name, **MacCartridge**, was unacceptable to Apple, who claimed that "Mac" infringed on their copyrighted trademark.

It's this idea of copyrighted products which I'm concerned about. The **Magic Sac** will be used by a number of people as a way to run unpurchased Macintosh programs on an ST. Data Pacific has taken extraordinary steps not to provoke Apple. The former does *not* sell official Apple Mac ROMs, will *not* tell you in person (or in print) where to get the ROMs, and does *not* publish a list of programs that work with **Magic Sac**. Further, Data Pacific's ads and promotional literature state plainly that the expected use of **Magic Sac** is for Apple Macintosh owners to run programs on their STs.

But, what Data Pacific carefully says publicly concerning the product's intent does not necessarily reflect how the product will be used. Some people will go to their local Apple dealer, buy the ROMs for about $30.00, buy the **Magic Sac** and then get copies of Mac disks from their friends.

A more serious issue than the potential pirating of Apple software is the possibility of less ST software. If enough ST owners find Mac programs that suit their needs, there may be a lessened demand for ST programs. With less users buying ST programs, less ST software may be produced.

It's often said that software piracy is responsible for the lack of new 8-bit Atari software. The last thing we as ST users need now is *another* reason for the software publishers *not* to want to produce Atari software.

I wonder if **Magic Sac** is the product the Atari ST community needs right now. Moreover, is it a product that the still-young personal computer industry needs right now? We'll never have lower software prices, as long as a software publisher has to figure the loss of revenue due to lost sales into his cost of doing business.

The **Magic Sac** is a neat accomplishment by David Small. He deserves kudos for his technical prowess. But I'm very concerned that this product may severely impact the future of the Atari ST—or other computers. A direct result of the **Magic Sac** could be less software for the ST. That would be a shame, given the increasing momentum earned by the ST computers. //

Ian's Quest

ST news, information and opinion

by Ian Chadwick

"Hello, I'd like to buy a copy of **MacPaint**," I said to the smiling, nattily dressed Apple dealer. Behind him, a group of programming types were playing with a resource editor on a Mac, and muttering about event timers and the like. I could tell they were programmers, because they were all drinking Coke and eating donuts. Real health food fanatics.

"Didn't you get one when you bought your Macintosh?" Smiley asked me somewhat condescendingly.

"It's not for a Mac," I replied. "It's for an ST."

"An ST? An *Atari* ST?" He dripped sarcasm all over the orange indoor-outdoor carpeting. One of the programming types overheard and rolled his eyes. Another sniggered. Gad, how I dislike people who snigger.

"Yes, an Atari ST. I'd like to buy **MacPaint** for my Atari ST. How much is it?"

"Sir," said Smiley, leaning forward and still dripping, "**MacPaint** will only run on a Macintosh. It will *not* run on an Atari of any sort, no matter how much they tried to make it look like a Macintosh." The sniggerer progressed to guffaws. Gad, how I dislike people who guffaw.

"No, it'll run on my ST, don't worry," I assured him, then added confidentially, "I have the monochrome monitor."

Smiley straightened up and looked down at me, frowning. "I don't know what those Atari dealers have been telling you, but under no circumstances will an Apple program designed for the Macintosh run on an Atari product." I could hear him capitalize words like *Apple* and *Macintosh*, just as if he were saying "God" and "the Constitution." Little icicles hung from every word. Probably trained in selling at Tuktoyaktuk U. I brushed the frost from my jacket and persisted.

"Yes it will. Listen: it's a 68000, right? So it will work. How much do you sell it for?"

Smiley wasn't smiling anymore. My insistence in wasting his valuable time was turning him frigid. Maybe I should have called him Frosty instead. "Sir, if you don't mind, I have more important . . ."

"Listen, I'll tell you what," I said easily. "I'll make you a bet. I have my ST in my car. I'll bring it in and load a copy of **MacPaint**. If it doesn't work, I'll give you a hundred dollars. If it does, you give me the program free of charge."

Frosty's eyes lit up like little squinty LEDs. "Fish in a barrel," he must have thought. "A fool and his money!" The programmers at the Mac station were grinning like predatory weasels, looking at me as if I were a trapped, fat chicken. I smiled at them, and I'll swear one began to salivate. Nice bunch, but would you want your sister to marry one? Frosty quickly agreed, and I dashed out to my car. Minutes later, I was struggling through the door with an armload of Atari equipment.

By now, several customers and the store manager had heard about me, and were all waiting around for my arrival. I think there were even bets that I wouldn't return. I fooled them. Mister Manager patted Frosty on the back (who—unable to decide which personality to wear—was being Smiley again). Everyone was enjoying the joke, at my expense. Smiley promised to buy Mr. Manager lunch when I'd gone. "Pick anywhere you want to eat," he said, "It's on me." Good career move.

I set up the ST, and hooked in the cables, power supplies and drives. The ST looks as if it rests on a Sargasso Sea of wires, compared to the Mac. I quietly plugged in my **Magic Sac** and booted the Atari. Several disparaging remarks were made about the screen and the menus by the programmer types. Mister Manager was loudly pointing out the mental deficiencies of the dealer who told me an ST would run Apple programs, counterpointing his arguments with his own store's virtues. I opened the **Magic Sac** loader, set it to 512K Motivator and, when it asked, I put in my special Finder 4.1 disk.

Clump clump clump clump—I heard the jaws dropping as the *Welcome to Macintosh* screen appeared. Someone muttered "good fake." Then the desktop cleared, and the whimpers of astonishment were audible as I quickly ran through the menu bar. I already had a copy of **MacPaint** on a **Magic**-formatted disk, so I put it in drive B and loaded the program. The blood had drained completely from Smiley's face. He was now Whitey. Mister Manager was glaring at him with eyes like little black holes. No one chortled even once.

Suddenly the customers were all over me, asking questions. I quit **MacPaint** and loaded **Excel**, then several public domain games from CompuServe's MACUS SIG. The programmers who recognized them were in awe. One kept mumbling "impossible" over and over again. When I booted **MDS Edit** from the Mac developer's kit, I was sure Whitey was going to faint. Especially when his boss said he wanted lunch at Maxim's. In Paris. France.

I answered a few questions—several from the customers, about the location of the closest Atari dealer—and demonstrated a few odds and ends, impressing everyone with the 30-percent increase in speed which **Magic Sac** has over the Mac. Then I closed it all down and stood up.

"Can I have my **MacPaint** now? I have to run." I said. Whitey was trembling. He pointed at the now dead screen.

"But you already have one. I saw it."

"Sure, but that's a copy. I want an original, so I can't be accused of piracy. I have an original of all the others. I just need **MacPaint**." I waited patiently, while Whitey and Mister Manager disappeared into the back room to retrieve a copy, probably from some carton full of Mac. When Whitey returned, I was all packed up and ready to go. He looked somewhat dishevelled, as though someone had brushed his suit with a live bobcat. Mister Manager didn't return, but I could hear him in the storeroom tearing walls down. Whitey gave me my **MacPaint** and I left. When I looked back, the programmer types were staggering around the store, looking as if someone had emptied a Buck Rogers stun gun on them. I smiled. I'd always wanted to do that.

Okay, okay, so the real secret lay in the **Magic Sac** (formerly the **MacCartridge**—see issue 45 of **ANALOG Computing** for the story of its introduction at the West Coast Computer Faire), from Data Pacific, Inc. It's a plug-in cartridge, equipped with two Apple Mac ROMs. I didn't tell them that—I couldn't resist the chance to pull off my trick. However, aside from playing practical jokes on Apple dealers, of what use is the **Magic Sac**?

Well, first of all, as advertised, it doesn't come with those very necessary ROMs. You need to pry them out of your Mac—and they must be *old* 64K ROMs, not the newest Mac Plus ROMs. Which makes this the most expensive cartridge in computer history: $129.00 for the cartridge and $3000.00 for the Mac (which doesn't work anymore, because you've just taken the ROMs out . . .)! DPI had to go this route, because Apple threatened legal action otherwise (the **Magic Sac** is sold as a "Mac" peripheral). Apple sues everyone. If you've seen the new GEM 2 from DR, you'll know how Apple's lawsuits can cripple a product. The company refuses to sell old ROMs to us Atari folk, even if the ROMs are useless to them. Apple doesn't like entrepreneurs to tread on their turf, either.

However, as I've discovered, a few enterprising local dealers have managed to obtain them for sale. I don't know where they got them—Apple says *they* won't sell them, but someone obviously does. The **Magic Sac** won't work with EPROMS. The sidebar to that story: Atari dealers can't sell any Mac software with the **Sac**—like the **System** and **Finder**. You still need to port those over from a Mac, using an ImageWriter cable. This assumes you have a Mac already, right? Otherwise, it's piracy. . .

Okay, so I have a Mac and an ST, and legit copies of everything (I really had **MacPaint**, too, but I just wanted to be mean . . . I find most Mac users insufferably arrogant). And if I port over the programs to my ST, is it piracy? Urk. Sticky question. Lots of people (and publishers) say yes. But it's not that easy.

First, if you take the ROMS out of your Mac, you're still really using the software on your Mac, or a part of it, 'cause the original ROMS are in use. So that can't be piracy, even if an ST is the transport for the data. Also, Mac disks won't load from the ST drives (at least until Data Pacific releases its Mac drive for the ST!) Protected software won't run, even when ported over, because, when it checks the drives for the protection, it isn't there on the ST drives. Lots of other programs won't run, either—including **MacWrite** (it will load and scroll, but touch the RETURN key, and—wham!—bomb city). And the music is permanently disabled. So, despite all the hoopla, there isn't an infinite number of programs that will work with it.

Many programs can't take advantage of the larger screen, although you can often use the space to open more windows, at least increasing their visibility and readability. But **MacPaint** still uses the teeny screen size, despite the largely unused border around it. Then you need an ImageWriter printer to dump **MacPaint** files. If you're planning to buy the Mac-compatible drive and the ImageWriter, why not just use the Mac? Sure, with the Mac drive they're planning, it'll be a lot cheaper than buying a real Mac. But how many Mac programs do I *really* want? **Excel**, **Think Tank**, a game or two. Not enough to make it worth the dollars for the extra hardware, at least, not personally.

I'd rather see the software manufacturers wake up to the growing sales of the ST and decide to port over their Mac products. Then I'd only need to buy the software. But that would require the sleepy industry—so long lulled by the IBM/clone and Apple sales—to realize that, gee whiz, there's something big happening out here. Don't hold your breath waiting. Lots of those folks are still saying, "Atari *who*?" when you try to discuss the ST with them. Sigh.

Speaking of software, there are two programs for which I'm still searching. Oh, variations exist, but none that meet my standards or needs: a good word processor and a good outline generator. I'm still using **ST Writer** (version 1.50 now. . .), because it does more for me than any of the others do. As for outline processors. . . well, let's leave these two for next month. Maybe by then I'll have some new products to discuss.

Ian Chadwick is a member of the Romance Writers of America and recently submitted a romance/mystery to a large paperback publisher. In his spare time, he writes mystery and spy fiction, collects rejection slips, walks the dog and ponders the meaning of existence.

DEGAS Elite

BATTERIES INCLUDED
30 Mural Street
Richmond Hill
Ontario, L4B 1B5 Canada
(416) 881-9941
520ST/1040ST $79.95
Upgrade for DEGAS owners $39.95
Not compatible with TOS in RAM

by Steve Rehrauer

I have a confession to make. For several months now, while you less fortunate souls struggled with your no-frills paint programs, I've been pampered. I've felt guilty cranking out hassle-free artwork when others must still sweat. (But, as the actress in the yuppie ice-cream commercial defiantly proclaims, I'd do it all again!)

Well, we can all rest easy now; **DEGAS Elite** is here at last. No greenhorn in the ST artware world, **Elite** boasts an impressive pedigree. Its parent program, Tom Hudson's **Design and Entertainment Graphics Arts System**, has already become an ST best-seller for its publisher, Batteries Included. Will the budding prodigy make "Daddy" proud? Listen, if this product were an automobile, Rolls Royce would be very, very nervous!

Trying to condense the sum of a product into a few pages is not an easy chore. And **Elite** is *big*. I can't hope to cover everything here, so I'll concentrate on the features not found in the original. Readers unfamiliar with **DEGAS** will bear with me, I hope.

The manual has been nearly doubled in length, and, like the first, it's well written. Unlike **DEGAS, Elite** may force you to keep the manual at hand. In spite of a very well designed user interface, **Elite** simply offers so many options that it isn't always apparent how to do what you want. You'll find yourself needing the keyboard more often, for example, especially the ESC, SHIFT, CTRL and ALT keys. I'm often confused as to which key to use, when. Thank goodness for manuals that have an index!

Several things stand out the first time you see **Elite**'s menu. For one, there's now a menu bar across the top. Drop-down menus offer an alternate way to select some functions that also appear on the menu, as well as some important new features that don't appear elsewhere.

Other changes stand out, like scroll arrows on the FILL, TEXT and LINE boxes. No more cycling through every available style to reach the one you want. But wait; what're those numbers in boxes at the bottom of the menu? Could it be. . .yes! Multiple workscreens! Click on the number of the picture you wish to work on; a standard 520ST with nothing else in memory (like a RAMdisk) can support two workscreens, while a 1040ST or 520ST with a megabyte of RAM can use up to eight. Pictures can be loaded into or saved from any workscreen.

But there's far more lurking beneath that deceptively **DEGAS**-like exterior. Snooping through the drop-down menus gives you an idea of what's been added.

(1) **Elite** can load picture files in any of the three **DEGAS** formats (.PI1, .PI2, .PI3), in **Neo-Chrome** format, in **KoalaPad** format (pictures ported from an 8-bit Atari), or even Amiga .1FF format. **Elite** will convert the picture to whatever resolution you are working in—amazingly well. Files can only be saved in **DEGAS** formats, however (except for block-images; more on those later).

(2) Picture files can be saved in standard **DEGAS** format, or in a compressed format. The savings vary from picture to picture. I find that compression typically makes my files about 30 percent smaller.

(3) A feature called ANIMATION lets you use four "animation channels" to do color cycling. In the low-resolution mode, each channel can consist of from one color to the entire palette of sixteen.

Cycling rate and direction through the palette are user selectable in **Elite**. The very slowest animation rate takes many seconds per color change, while the fastest blurs all colors in the group into a single, flickering color.

Picture files saved with **Elite** have extra information appended to describe the animation channels. This not only preserves animation effects for the next session, but is used by the improved "picture showcase" program (SHOWPIC2) provided on the disk.

You can use drives A: through P: to save pictures, so hard-disk users (or those with enough memory for a RAMdisk with **Elite**) can dramatically cut the time they spend waiting for disk operations.

(5) A powerful new feature, "block operations", has replaced the COPY and MOVE functions of **DEGAS**. Any portion of a picture can be "grabbed" and dumped into a "block buffer," where it can be rotated, stretched or distorted in a variety of ways. This picture chunk can then be used as the cursor for nearly all the normal operations (DRAW, POINT, BOX, etc), which is how one can easily copy and move portions (or all) of a picture between the workscreens. Blocks may also be saved to or retrieved from disk, allowing you to create your own "clip art" library.

And there's much, much more.

COMPUTER CREATIONS

YOUR ATARI 520ST SUPPORT CENTER

ABACUS	
Paint Pro	36
Text Pro	36
Data Trieve	36
ACADEMY	
Typing Tutor/ Word Invaders	23
ACCESS	
Leaderboard Golf	26
Tournament Disk	15
ACCOLADE	
Mean 18 Golf	29
Sundog	26
ACTION	
Action Pak	32
ACTIVISION	
Mindshadow	33
Borrowed Time	33
Hacker	29
Music Studio	39
Paintworks	47
Little Comp. People	33
Hacker II	33
Tass Times/Tonetown	33
Champ. Basketball	29
ADVENTURE INTL.	
Spiderman	14
ANTIC	
C.O.L.R. Obj. Editor	23
Maps & Legends	27
Macro Assembler	60
Lattice C	114
Disk Doctor	23
A-Calc	45
Meta Pascal	75
CAD-3D	38
A-Ram	15
A-Seka	27
GST C Compiler	60
GST-ASM	45
Expert Opinion	75
Flash	30
Star Struck	19
Red Alert	19
Murray and Me	15
Mom and Me	15
Kermit & Remote Co.	19
Crystal	19
ARTWORX	
Bridge 4.0	19
Compubridge	19
Maillist	12
Strip Poker	25
Peggammon	12
Hole in One Golf	19
ATARI	
DB Man	103
DB Master	34
Joust	21
Star Raiders	21
BATTERIES INCLUDED	
Degas	26
Degas Elite	49
Timelink	33
Thunder	26
I/S Talk	33
I/S Paperclp Elite	Call
Homepak	33
BECKEMEYER	
Micro C-Shell	39
Micro C-Tools	19
Micro Make	27
RTX	53
Micro MTC Shell	79
Hard Disk Accelerate	30
BRODERBUND	
Mindwheel	29
Breakers	30
CENTRAL POINT	
Copy II	27
CLASSIC IMAGE	
Disk Library	38
Diablo	23
COSMI	
Super Huey	30
DAC	
Easy Accounting	49
DRAGON GROUP	
4 x Forth	79
Forth Accelerator	59
ELECTRONIC ARTS	
Financial Cookbook	36
Ultima III	39
Coloring Book	Call
Golden Oldies	23
Chessmaster 2000	Call
OGRE	Call
Autoduel	Call
Skyfox	Call
EPYX	
Winter Games	26
Rogue	26
Temple of Apshai	26
World Games	26
Champion. Wrestling	26
Super Cycle	26
FIREBIRD	
The Pawn	30
Starglider	30
FIRST BYTE	
Kid Talk	33
Speller Bee	33
Math Talk	33
First Shapes	33
HABA	
Habacom	19
Phonebook	39
Home Accountant	51
Mailroom	56
Checkminder	49
HIPPOPOTAMUS	
Hippoword	59
Hippoconcept	59
Hipposimple	33
Hippo Disk Utilities	33
Hippo Backgammon	27
Hippo Spell	27
Hippo Ramdisk	23
Hippo Comp. Almanac	23
Hippo Jokes & Quotes	23
Hippo Art I	27
Hippo Pixel	27
Hippo Fonts I	27
Hippo Laser	47
Hippo Eprom Burner	109
Hippo Sound Digitz.	109
Hippo Vision B & W	109
HOLMES & DUCKWORTH	
Tool Box	28
LOGICAL DESIGN WORKS	
LDW Basic Compiler	54
MARK OF THE UNICORN	
Final Word	99
Hex	27
PC/Intercomm	84
MEGASOFT	
A Copier	27
X-Rated Library	17

MICHTRON	
Kissed	26
DOS Shell	26
DFT	33
The Animator	26
Personal Money Mgr	33
Utilities	39
M-Disk	26
Mudpies	26
Softspool	26
Flip Side	26
Calendar	19
MI-Term	33
Gold Runner	26
Time Bandits	26
BBS	33
Cornerman	33
Mighty Mail	33
Lands of Havoc	13
Cards	26
Business Tools	33
Logo	33
Major Motion	26
Mission Mouse	26
MI-Dupe	20
BBS 2.0	54
Pro Football Wizard	27
Pinball Factory	27
Your Finance Future	26
MICROPROSE	
Silent Service	27
MI-GRAPH	
Easy Draw	54
MINDSCAPE	
Brataccus	33
NAVARONNE	
Astrology	29
Timekeeper	37
OMNITRENDS	
Universe II	47
OSS	
Personal Pascal	50
OTHER VALLEY SOFTWARE	
Delta Patrol	17
Monkey Business	17
PENGUIN/POLARWARE	
Crimson Crown	15
OO-Topos	15
Sword of Kadash	15
The Coveted Mirror	15
Transylvania	15
PHILON	
Fast Basic-M	99
PROCO	
Pro-Copy	19
PROGRESSIVE COMP. APPL.	
Graphic Artist	139
Font Editor	54
PRYORITY	
Forbidden Quest	27
QUANTUM	
ST-Talk	14
QUICKVIEW	
Zoomracks	54
REGENT	
Regent Spell	34
Regent Word	34
ROYAL	
EZ-Calc	54
SHANNER	
Soundwave SW-1	33
ST-Key	14
Macro-Manager	33
DB Calc II	29
SHELBOURNE	
Pool	24
SIERRA	
Kings Quest II	33
Ultima II	39
Winnie the Pooh	19
Black Cauldron	26
K/Q Hint Book	6
B/C Hint Book	6
Donald Duck	19
Cash Disbursements	67
SPINNAKER	
Homework Helper:Math	34
Treasure Island	27
SST	
Chat	15
SUBLOGIC	
Flight Simulator II	38
TDI	
Modula-2	59
Modula-2 Develp.Kit	109
UCSD Pascal	59
TELARIUM	
Amazon	Call
Nine Princes/Amber	Call
Farenheit 451	Call
Dragonworld	Call

TIMEWORKS	
Datamanager	54
Swiftcalc	54
Wordwriter	54
S/P Financial Plan	54
UNICORN	
Fraction Action	27
Decimal Dungeon	27
UNISON WORLD	
Printmaster	26
Art Gallery I	19
Art Gallery II	19
VIP	
VIP Professional	119
X-LENT	
Typesetter Elite	29
Rubber Stamp	25
Music Box	32
Megafont	25

PRINTERS

OKIDATA	
Okimate 20 & Plug N' Print	219
STAR MICRONICS	
LV-1210	189
NX-10	259
PANASONIC	
KX-P 10801	199
KX-P 10911	259
KX-P 3131	255
KX-P 1592	429
MODEMS	
Supra 300 ST	59
Supra 1200 ST	149
Avatex 1200	99
Avatex 1200 HC	139
Compuserve Start Kt	24
Microstuffer	59
Omega Terminal	20

ST ACCESSORIES

Disk Cleaning Kit	14
Dust Covers	7
Monitor Stand	12
Mouse Pad	8
Shanner Planner	29
The Easel	14
Disc Directory	20
Pocket Pak	10
The Library	34
Speed Pad	18
Disk File/30 Micro	10

ST BOOKS

Atari ST GEM Reference
Atari ST Internals
Atari ST Machine Language
Atari ST Tips & Tricks
Atari ST from Basic to C
Atari ST Basic Training
Atari ST Graphic & Sound
Atari ST Logo
Atari ST Peeks & Pokes
Atari ST for Beginners

CABLES

ST to Printer	15
ST to Modem	19

To order call TOLL FREE

1-800-824-7506

ORDER LINE ONLY

No extra charge for MasterCard or Visa

COMPUTER CREATIONS, Inc.

P.O. BOX 493 - DAYTON, OHIO 45459

For information, order inquires, or for Ohio orders (513) 435-6868

VISA® MasterCard

Order lines Open 9:00 a.m. to 8:00 p.m. Mon.-Fri.; 10 a.m. to 4:00 p.m. Sat. (Eastern Standard Time). Minimum $15 per order. C.O.D. (add $3.00). Please specify computer system. Call toll free number to verify prices and availability. Prices and availability are subject to change without notice. We ship C.O.D. to Continental U.S. addresses only! Please include 4% shipping on all Hardware orders (min. $4.00). Software and accessories add $3.00 shipping and handling in continental U.S. Actual freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico and APO. Ohio residents add 6¼% sales tax. Canadian orders add 5% shipping, (min. $5.00). All other foreign orders, please add 15% shipping, (min. $10). For immediate delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 3 weeks to clear. School purchase orders welcome. Due to our low prices, all sales are final. NO CREDITS. All defective returns must have a return authorization number. Please call (513) 435-6868 to obtain an RA# or your return will not be accepted for replacement or repair. FOR YOUR PROTECTION WE CHECK FOR CREDIT CARD FRAUD. We do not bill until we ship.

CIRCLE #118 ON READER SERVICE CARD

The keyboard can be used to perform all drawing functions, as an alternative to the mouse and/or drop-down menus. You can even use the keyboard to move the mouse cursor, in cursor-length or single-pixel distances at a stroke.

All the fill patterns, lines and brushes can be customized (in **DEGAS**, only one of each was changeable). Better yet, fill patterns can use multiple colors. In a really neat touch, any portion of a picture can be grabbed to become a fill pattern.

There are several "modes" which affect most of the drawing functions in **Elite**:

NORMAL — Just as it sounds. Everything behaves as in **DEGAS**.

CHANGE — Lets you substitute one color in a picture for another. This works with most modes, and is especially handy with patterns or intricate detail. Instead of doing a pixel-by-pixel change under the magnifier, just SET CHANGE, select the color you wish to change, select the color to replace it, and pick one of the drawing functions (a fat brush with DRAW works fine). Only the color you're changing will be affected; the surrounding parts of the picture aren't touched.

CYCLE — This affects only DRAW, and works especially well with ANIMATION. Essentially, CYCLE changes the color of your brush as you draw with it. The colors used are those in the currently-selected animation channel. With ANIMATION off, you'll just get "banded" lines when you draw. But, when used with ANIMATION, the bands appear to ripple along the line, creating a "marquee light" effect.

PATTERN — This mode, instantly familiar to **MacPaint** users, lays down the current fill pattern. Unlike the regular FILL, it lets you DRAW with patterns, or even AIRBRUSH them. And, if there are blank areas in the pattern, (like spaces between cross-hatching), whatever was already on the picture will show through there.

SMEAR — Just as it sounds. Imagine running your fingertip over a wet painting, and you'll get an idea of what SMEAR can do.

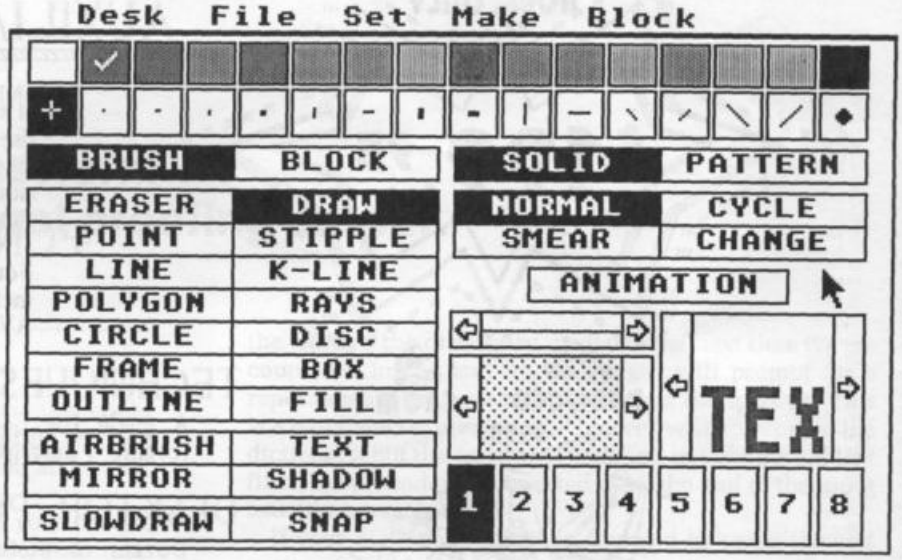
There are other differences. The magnifier has been improved substantially. For starters, you can choose how much magnification to use, ranging from 3X through 12X. When the magnified portion appears, you'll see what I find the biggest improvement: a direction-box with scroll arrows for up, down, left and right, to scroll the magnifier across the full picture. To the left of the magnified pixels is a box with an actual size portion of the full picture, with the current position of the magnifier on it. Along with the scroll arrows are controls to "zoom" in or out on the magnified pixels, within the 3X-12X range.

A STIPPLE function has been added. It's very similar to AIRBRUSH, the difference being that STIPPLE places "dots" in the shape of the current brush.

A SNAP function lets you turn on an invisible grid over your picture. Everything is then constrained to land on a grid-point. This is handy for ensuring proper alignment, especially in high resolution.

An OUTLINE function outlines an object with a specified color.

New SET COLOR features have been added. You can easily "copy" colors in the palette, by simply selecting a color box and "dragging" it onto another.



Elite automatically generates the spectrum of shades between two colors for you. Select one color, hold down the ALT key and click on another color box—all the color boxes between these two will be filled with smoothly-gradated shades of the chosen colors. You can point to a pixel on a picture and have **Elite** tell you which color box holds that color. And you can view the entire ST range of 512 colors at once, choosing your palette from that.

One can see an example of **Elite**'s polish by converting a high-resolution monochrome picture to low resolution. Not only will **Elite** convert the picture to the lower resolution, but it remaps the current color palette, to provide gray scales about as close to the original monochrome picture as one could ask for.

Another example. Although **Elite** allows multiple workscreens, all must share the same color palette. When loading a picture, unless the current palette and the palette in the picture file are the same, the loaded picture could look awful.

So this wouldn't be a useful feature, were it not for one thing. When **Elite** loads the image, it asks you if you want to apply the current palette (and possibly get the awful results) or recolor the image with the current palette (**Elite** calls this remapping). If you choose remapping, **Elite** finds the best fit between the current palette and the saved picture's palette, and changes the image accordingly. If the image has a red color, **Elite** picks the closest thing to that red in the current palette and recolors the red portions of the loaded image with that color. It takes a few seconds to do, but the results are usually very good.

Now, every reviewer has to have at least a few gripes with a product, right? Those who don't probably never used the thing in the first place. Well, uh, I've been using **Elite** for quite a while, and I *really* can't find much to complain about.

Oh, if you prodded me hard enough, I suppose I could fine a few things. **Elite** is just a hair more obtuse to use; power has a price, and there's only so much that will fit onto a menu screen. Also, chalk it up to my own clumsiness, if you like, but I have difficulties rotating an object; the object tends to skitter away from the cursor when I try grabbing a corner of it.

I wouldn't mind having one more block option that would stretch the image by the same amount in both horizontal and vertical directions. And a "yardstick" feature, to measure the dimensions of things in terms of pixels; that'd be handy sometimes.

But all that's nit-picking. The fact is, this is an amazing program.

Batteries Included offers an upgrade for current **DEGAS** users. Is **Elite** worth the trade of your master disk plus $40.00? For color systems, my response is an unqualified yes. Monochrome users are a bit shortchanged, but I'd still consider the upgrade a good deal. For my money, multiple workscreens, block operations and an improved magnifier are easily worth it.

And, for those of you still waiting to buy ST artware, your choice has just been narrowed—to buy **DEGAS Elite**, or not to buy at all.

the Graphic Artist®

The *first* software to combine

Computer Aided Design
and
Desktop Publishing

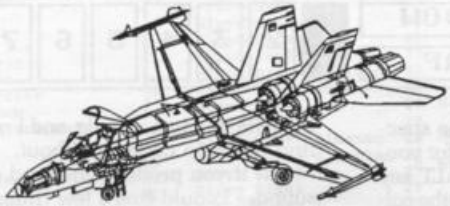
in a single environment.

WRITTEN and DESIGNED By Peter Naleszkiewicz

NOW ONLY

$199.95

REDUCED FROM $495.00



"The GRAPHIC ARTIST is one of the first programs that tries to take advantage of the tremendous power of the ST ... the first ST program to use the printer to its highest resolution and the results are extremely impressive ... The GRAPHIC ARTIST can indeed produce professional drawings ... quite an impressive product."

- Current Notes

"PCA's GRAPHIC ARTIST is a very sophisticated program that manipulates graphics and text for sending to a plotter, laser printer or dot matrix printer. GRAPHIC ARTIST has professional level ambitions in computer aided design (CAD), business graphics and desktop publishing. You can create some great on-screen graphics with the GRAPHIC ARTIST, but the program's real purpose is to print its end results..."

- Antic Publishing

AUXILIARY SOFTWARE:

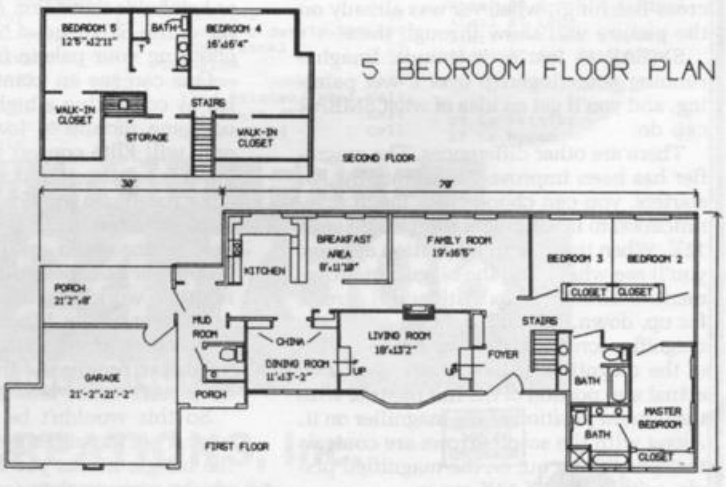
FONT EDITOR $79.95
Use to create new Graphic Artist fonts (like the one you are reading), or modify existing ones. AVAILABLE NOW.

FONTPACK-1 $49.95
Four additional font styles, including the two used in this ad. AVAILABLE NOW.

TECHNIQUES and TIPS $39.95
A guide for both beginner and expert. Includes file format descriptions. AVAILABLE IN NOVEMBER.

DESKTOP PUBLISHING LIBRARY $49.95
Dozens of useful pictures as symbols. IN NOVEMBER.

ELECTRONICS LIBRARY $49.95
A set of electronic symbols. IN DECEMBER.



5 BEDROOM FLOOR PLAN

** THIS ENTIRE AD WAS DESIGNED WITH AND PRINTED BY THE GRAPHIC ARTIST and FONTPACK-1 USING AN HP LASERJET PRINTER. Shown ACTUAL size. **

Progressive Computer Applications Inc.

2002 McAuliffe Dr. Rockville, MD 20851
(301) 340-8398

CIRCLE #119 ON READER SERVICE CARD

Spellbinder

Create dictionaries, perform real-time editing and get proofing output.

by Kirk Stover

Have you ever wished for a spelling checker that could proofread any text file, allowing real-time editing and dictionary updates? Well, **Spellbinder** offers you all of that, and more!

Spellbinder reads through the text file of your choice, creating or updating a dictionary for later use, and creating a proofread output file. It's very easy to use. To make your copy of **Spellbinder**, type in the ST BASIC program shown in Listing 1 (check your typing with **ST Check**—see page 84) and run it. The file SPELLER.TOS will be written to your disk. Listing 2 is the source code for **Spellbinder** and is not necessary to run the program. It is included for those who have an interest in 68000 assembly language.

To run the program, just double click on SPELLER.TOS. This will present you with a prompt that asks you to type in the name of the dictionary to be used. If you enter a dictionary name which already exists, it will be loaded into memory. Otherwise, you'll be given the option of retrying, or creating a new dictionary file. Disk subscribers will find a beginning dictionary on this issue's disk, and it will also be available on the Atari Users' Group on Delphi.

Once the dictionary name has been established, **Spellbinder** will ask you the name of the text file to be proofread. Simply type in the filename. If the program doesn't find it, you can either retry, or cancel and return to the desktop. The final prompt will ask for the name of the proofread output file.

The program will begin reading the input file, attempting to match the words with those found in the dictionary. You can watch the program's progress on-screen. Anytime it reads a word not found in the dictionary, that word will be displayed in reverse video, and you'll be given the option to add, change, or ignore it. "Add" will place the word in the dictionary, available the next time it's encountered in the text file. "Change" will prompt for a replacement string, so you can correct the spelling. This is a 64-character maximum. "Ignore" will not update the dictionary, but the word will be found in your output text file. This procedure is repeated until the end of the input text file is reached.

If your dictionary has been changed by using the add option, **Spellbinder** will ask if you want to save the updated version of the dictionary. Hitting *Y* will bring a prompt for the new dictionary name, then an attempt to save it with your changes. It's a good practice to always use a new name, so you have a backup copy. Pressing *N* will take you straight to the "thank you" sign-off without changing the dictionary.

Helpful hints.

If you have multiple drives or a RAMdisk, you can speed up the process, by specifying different devices for the filenames. The larger a dictionary is, the slower its reading will go—and the likelihood of finding matches will also be increased.

Reading several text files will create a dictionary of the words you'll most often encounter. The dictionary files you create can be edited with any standard text editor, and can be printed by simply double clicking on the filename on the desktop. This will produce an easy-to-read dictionary listing, with each word on its own line.

Spellbinder allocates memory for the dictionary at run time. If you get "load" errors when loading the dictionary, rename your desk accessory files and/or RAM disk, and reboot your system.

Kirk Stover is a Systems Analyst at an insurance company in Minnesota. He enjoys working on his 520ST in both C and assembly language. His special interest is writing time-saving utilities.

Listing 1.
ST BASIC listing.

```
100 filename$="a:\SPELLER.TOS"
110 fullw 2:clearw 2:gotoxy 0,0:print
"creating file..."
120 option base 0
125 dim a%(16000):def seg=i:v$=""
130 p=varptr(a%(0)):bptr=p+1
140 for i%=1 to 3065
150 read v$:code%=val("&H"+v$)
160 poke p, code%:print ".";
170 p=p+1
180 next
190 bsave filename$,bptr,3065
200 print "file written":end
1000 data 60,1A,00,00,0B,78,00,00,00,0
0,00,00,00,00,00,00
1010 data 00,00,00,00,00,00,00,00,00,0
0,00,00,2A,4F,2A,6D
1020 data 00,04,20,2D,00,0C,D0,AD,00,1
4,D0,AD,00,1C,06,80
1030 data 00,00,01,00,2F,00,2F,0D,3F,3
C,00,00,3F,3C,00,4A
1040 data 4E,41,DF,FC,00,00,00,0C,61,0
0,00,26,61,00,00,6A
1050 data 61,00,01,38,61,00,01,88,41,F
9,00,00,08,2F,61,00
1060 data 05,C6,61,00,01,CE,61,00,04,7
8,3F,3C,00,00,4E,41
1070 data 13,FC,00,00,00,00,0A,DB,23,F
C,00,00,00,00,00,00
1080 data 0A,E0,2F,3C,FF,FF,FF,FF,3F,3
C,00,48,4E,41,5C,8F
1090 data 23,C0,00,00,0A,D6,2F,00,3F,3
C,00,48,4E,41,5C,8F
1100 data 23,C0,00,00,0A,DC,23,C0,00,0
0,0A,E4,41,F9,00,00
1110 data 06,18,61,00,05,72,4E,75,41,F
9,00,00,08,D5,61,00
1120 data 05,12,3F,3C,00,00,2F,3C,00,0
0,0A,AE,3F,3C,00,3D
1130 data 4E,41,50,8F,4A,40,6A,00,00,2
A,41,F9,00,00,09,60
1140 data 61,00,05,44,41,F9,00,00,0A,8
5,61,00,05,3A,61,00
1150 data 05,18,0C,00,00,72,67,C0,0C,0
0,00,63,66,E6,60,00
1160 data 00,88,33,C0,00,00,0A,A6,41,F
9,00,00,09,2F,61,00
1170 data 05,16,41,F9,00,00,0A,8E,61,0
0,05,0C,2F,39,00,00
1180 data 0A,DC,2F,39,00,00,0A,D6,3F,3
9,00,00,0A,A6,3F,3C
1190 data 00,3F,4E,41,DF,FC,00,00,00,0
C,4A,80,6B,00,00,0C
1200 data B0,B9,00,00,0A,D6,66,00,00,2
0,41,F9,00,00,09,B9
1210 data 61,00,04,D4,41,F9,00,00,0A,8
5,61,00,04,CA,61,00
1220 data 04,A8,58,8F,60,00,FF,04,23,C
0,00,00,0A,E0,D0,B9
1230 data 00,00,0A,DC,23,C0,00,00,0A,E
4,3F,39,00,00,0A,A6
1240 data 3F,3C,00,3E,4E,41,58,8F,4E,7
5,41,F9,00,00,08,F6
1250 data 61,00,04,40,3F,3C,00,00,2F,3
C,00,00,0A,AE,3F,3C
1260 data 00,3D,4E,41,50,8F,4A,40,6A,0
0,00,2C,41,F9,00,00
1270 data 09,DE,61,00,04,72,41,F9,00,0
0,0A,85,61,00,04,68
1280 data 61,00,04,46,0C,00,00,72,67,C
0,0C,00,00,63,66,E6
1290 data 58,8F,60,00,FE,96,33,C0,00,0
0,0A,A8,4E,75,41,F9
1300 data 00,00,09,12,61,00,03,EC,3F,3
C,00,00,2F,3C,00,00
1310 data 0A,AE,3F,3C,00,3C,4E,41,50,8
F,4A,40,6A,00,00,2C
1320 data 41,F9,00,00,0A,0A,61,00,04,1
E,41,F9,00,00,0A,85
1330 data 61,00,04,14,61,00,03,F2,0C,0
0,00,72,67,C0,0C,00
1340 data 00,63,66,E6,58,8F,60,00,FE,4
2,33,C0,00,00,0A,AA
1350 data 4E,75,2F,3C,00,00,0A,F6,2F,3
C,00,00,00,01,3F,39
1360 data 00,00,0A,A8,3F,3C,00,3F,4E,4
1,DF,FC,00,00,00,0C
1370 data 0C,80,00,00,00,01,66,00,00,C
4,10,39,00,00,0A,F6
1380 data 0C,00,00,41,6D,00,00,1A,0C,0
0,00,5B,6D,00,00,44
1390 data 0C,00,00,61,6D,00,00,0A,0C,0
0,00,7B,6D,00,00,34
1400 data 2F,3C,00,00,0A,F6,2F,3C,00,0
0,00,01,3F,39,00,00
1410 data 0A,AA,3F,3C,00,40,4E,41,DF,F
C,00,00,00,0C,42,40
1420 data 10,39,00,00,0A,F6,3F,00,3F,3
C,00,02,4E,41,58,8F
1430 data 60,80,26,7C,00,00,0A,F6,52,8
B,2F,0B,2F,3C,00,00
1440 data 00,01,3F,39,00,00,0A,A8,3F,3
C,00,3F,4E,41,DF,FC
1450 data 00,00,00,0C,0C,80,00,00,00,0
1,66,00,00,3C,10,13
1460 data 0C,00,00,41,6D,00,00,16,0C,0
0,00,5B,6D,CA,0C,00
1470 data 00,61,6D,00,00,08,0C,00,00,7
B,6D,BC,13,C0,00,00
1480 data 0A,DA,16,BC,00,00,61,00,00,3
2,13,F9,00,00,0A,DA
1490 data 00,00,0A,F6,60,00,FF,6A,61,0
0,00,20,3F,39,00,00
1500 data 0A,A8,3F,3C,00,3E,4E,41,58,8
F,3F,39,00,00,0A,AA
1510 data 3F,3C,00,3E,4E,41,58,8F,4E,7
5,26,79,00,00,0A,DC
1520 data 45,F9,00,00,0A,F6,B7,F9,00,0
0,0A,E4,67,00,00,34
1530 data 16,1B,14,1A,0C,03,00,0D,66,0
0,00,0E,0C,02,00,00
1540 data 67,00,00,56,52,8B,60,D8,0C,0
2,00,00,67,00,00,0A
1550 data 00,02,00,20,B6,02,67,D8,10,1
B,0C,00,00,0A,66,F8
1560 data 60,BE,41,F9,00,00,0A,EE,61,0
0,02,9C,41,F9,00,00
1570 data 0A,E8,61,00,02,92,41,F9,00,0
0,0A,F6,61,00,02,88
1580 data 41,F9,00,00,0A,EB,61,00,02,7
E,61,00,00,4C,41,F9
1590 data 00,00,0A,F1,61,00,02,70,41,F
9,00,00,0A,F6,61,00
1600 data 02,66,41,F9,00,00,0A,F6,61,0
0,00,1E,2F,3C,00,00
1610 data 0A,F6,2F,00,3F,39,00,00,0A,A
A,3F,3C,00,40,4E,41
1620 data DF,FC,00,00,00,0C,4E,75,42,8
0,0C,18,00,00,67,00
1630 data 00,06,52,80,60,F4,4E,75,41,F
9,00,00,08,5D,61,00
1640 data 02,26,61,00,02,04,0C,00,00,6
1,66,00,00,0A,61,00
1650 data 00,28,60,00,00,18,0C,00,00,6
3,66,00,00,0A,61,00
1660 data 00,92,60,00,00,08,0C,00,00,6
9,66,CC,41,F9,00,00
1670 data 08,31,61,00,01,F2,4E,75,41,F
9,00,00,0A,F6,61,A8
1680 data D0,B9,00,00,0A,E0,06,80,00,0
0,00,02,B0,B9,00,00
1690 data 0A,D6,6D,00,00,14,41,F9,00,0
0,08,B3,61,00,01,C8
1700 data 61,00,01,A6,60,00,00,4A,13,F
```

```
C,00,01,00,00,0A,DB
1710 data 41,F9,00,00,0A,F6,22,79,00,0
0,0A,E4,10,18,06,B9
1720 data 00,00,00,01,00,00,0A,E0,0C,0
0,00,00,67,00,00,0A
1730 data 00,00,00,20,12,C0,60,E4,12,F
C,00,0D,12,FC,00,0A
1740 data 23,C9,00,00,0A,E4,06,B9,00,0
0,00,01,00,00,0A,E0
1750 data 4E,75,41,F9,00,00,08,9F,61,0
0,01,6C,13,FC,00,40
1760 data 00,00,0A,F4,2F,3C,00,00,0A,F
4,3F,3C,00,0A,4E,41
1770 data 5C,8F,41,F9,00,00,0A,F6,D0,C
0,10,BC,00,00,4E,75
1780 data 41,F9,00,00,06,18,61,00,01,3
E,0C,39,00,01,00,00
1790 data 0A,DB,66,00,00,C4,41,F9,00,0
0,0A,38,61,00,01,28
1800 data 41,F9,00,00,0A,85,61,00,01,1
E,61,00,00,FC,0C,00
1810 data 00,6E,67,00,00,A4,0C,00,00,7
9,66,DA,41,F9,00,00
1820 data 08,D5,61,00,00,AE,3F,3C,00,0
0,2F,3C,00,00,0A,AE
1830 data 3F,3C,00,3C,4E,41,50,8F,4A,4
0,6A,00,00,2A,41,F9
1840 data 00,00,09,8C,61,00,00,E0,41,F
9,00,00,0A,85,61,00
1850 data 00,D6,61,00,00,B4,0C,00,00,7
2,67,9A,0C,00,00,63
1860 data 66,DC,60,00,00,54,33,C0,00,0
0,0A,A6,41,F9,00,00
1870 data 09,48,61,00,00,B2,41,F9,00,0
0,0A,8E,61,00,00,A8
1880 data 2F,39,00,00,0A,DC,2F,39,00,0
0,0A,E0,3F,39,00,00
1890 data 0A,A6,3F,3C,00,40,4E,41,DF,F
C,00,00,00,0C,B0,B9
1900 data 00,00,0A,E0,66,98,3F,39,00,0
0,0A,A6,3F,3C,00,3E
1910 data 4E,41,58,8F,4A,40,6B,86,41,F
9,00,00,0A,61,61,00
1920 data 00,66,41,F9,00,00,0A,85,61,0
0,00,5C,61,00,00,3A
1930 data 4E,75,61,00,00,52,41,F9,00,0
0,0A,85,61,00,00,48
1940 data 13,FC,00,28,00,00,0A,AC,2F,3
C,00,00,0A,AC,3F,3C
1950 data 00,0A,4E,41,5C,8F,4A,40,67,D
C,41,F9,00,00,0A,AE
1960 data D0,C0,10,BC,00,00,4E,75,3F,3
C,00,FF,3F,3C,00,06
1970 data 4E,41,58,8F,4A,40,66,F0,3F,3
C,00,01,4E,41,54,8F
1980 data 00,00,00,20,4E,75,2F,08,3F,3
C,00,09,4E,41,5C,8F
1990 data 4E,75,00,00,1B,45,0D,0A,20,2
0,20,20,20,20,20,20
2000 data 20,20,20,20,2F,2D,2D,2D,2F,7
C,0D,0A,20,20,20,20
2010 data 20,20,20,20,20,20,20,2F,20,2
0,20,2F,20,7C,0D,0A
2020 data 20,20,20,20,20,20,20,20,20,2
0,2F,20,20,20,2F,20
2030 data 20,7C,20,20,20,20,20,20,20,2
```

```
0,20,20,20,20,20,20
2040 data 20,1B,70,20,20,20,20,20,20,2
0,20,20,20,20,20,20
2050 data 20,20,20,20,20,1B,71,0D,0A,2
0,20,20,20,20,20,20
2060 data 20,20,2F,20,20,20,2F,20,20,2
0,7C,20,20,20,20,20
2070 data 20,20,20,20,20,20,20,20,20,2
0,1B,70,20,53,70,65
2080 data 6C,6C,69,6E,67,20,43,68,65,6
3,6B,65,72,20,1B,71
2090 data 0D,0A,20,20,20,20,20,20,20,2
0,2F,20,20,20,2F,20
2100 data 20,20,20,7C,20,20,20,20,20,2
0,20,20,20,20,20,20
2110 data 20,20,20,1B,70,20,20,20,20,2
0,20,20,20,20,20,20
2120 data 20,20,20,20,20,20,20,1B,71,0
D,0A,20,20,20,20,20
2130 data 20,20,2B,2D,2D,2D,2B,20,20,2
0,20,20,7C,0D,0A,20
2140 data 20,20,20,20,20,20,7C,20,20,2
0,7C,20,20,20,20,20
2150 data 7C,0D,0A,20,20,20,20,20,20,2
0,7C,20,53,20,7C,20
2160 data 20,20,20,20,7C,0D,0A,20,20,2
0,20,20,20,20,7C,20
2170 data 50,20,7C,20,20,20,20,20,7C,0
D,0A,20,20,20,20,20
2180 data 20,20,7C,20,45,20,7C,20,20,2
0,20,20,7C,0D,0A,20
2190 data 20,20,20,20,20,20,7C,20,4C,2
0,7C,20,20,20,20,20
2200 data 7C,0D,0A,20,20,20,20,20,20,2
0,7C,20,4C,20,7C,20
2210 data 20,20,20,20,20,7C,0D,0A,20,20,2
0,20,20,20,20,7C,20
2220 data 42,20,7C,20,20,20,20,20,7C,0
D,0A,20,20,20,20,20
2230 data 20,20,20,7C,20,49,20,7C,20,20,2
0,20,20,7C,0D,0A,20
2240 data 20,20,20,20,20,20,7C,20,4E,2
0,7C,20,20,20,20,20
2250 data 7C,0D,0A,20,20,20,20,20,20,2
0,7C,20,44,20,7C,20
2260 data 20,20,20,20,7C,0D,0A,20,20,2
0,20,20,20,20,7C,20
2270 data 45,20,7C,20,20,20,20,2F,0D,0
A,20,20,20,20,20,20
2280 data 20,7C,20,52,20,7C,20,20,20,2
F,0D,0A,20,20,20,20
2290 data 20,20,20,20,7C,20,20,20,7C,20,2
0,2F,0D,0A,20,20,20
2300 data 20,20,20,20,20,7C,20,56,20,7C,2
0,2F,0D,0A,20,20,20
2310 data 20,20,20,20,20,7C,31,2E,30,7C,2
F,0D,0A,20,20,20,20
2320 data 20,20,20,20,2B,2D,2D,2D,2B,0D,0
A,00,1B,45,1B,48,1B
2330 data 6C,20,2A,2A,2A,20,53,70,65,6
C,6C,62,69,6E,64,65
2340 data 72,20,53,70,65,6C,6C,69,6E,6
7,20,43,68,65,63,6B
2350 data 65,72,20,2A,2A,2A,0D,0A,00,1
B,48,1B,6C,4E,6F,20
2360 data 6D,61,74,63,68,20,77,61,73,2
0,66,6F,75,6E,64,2E
2370 data 20,20,53,65,6C,65,63,74,20,6
F,70,74,69,6F,6E,20
2380 data 28,41,29,64,64,20,28,43,29,6
8,61,6E,67,65,20,28
2390 data 49,29,67,6E,6F,72,65,3A,20,2
0,00,1B,48,1B,6C,52
2400 data 65,70,6C,61,63,65,20,77,69,7
4,68,3A,20,20,00,1B
2410 data 48,1B,6C,4E,6F,20,6D,6F,72,6
5,20,72,6F,6F,6D,20
2420 data 69,6E,20,64,69,63,74,69,6F,6
E,61,72,79,21,20,20
2430 data 00,1B,59,29,3E,45,6E,74,65,7
2,20,44,69,63,74,69
2440 data 6F,6E,61,72,79,20,46,69,6C,6
5,20,4E,61,6D,65,1B
2450 data 4B,00,1B,59,29,3E,45,6E,74,6
5,72,20,49,6E,70,75
2460 data 74,20,46,69,6C,65,20,4E,61,6
D,65,1B,4B,00,1B,59
2470 data 29,3E,45,6E,74,65,72,20,4F,7
5,74,70,75,74,20,46
2480 data 69,6C,65,20,4E,61,6D,65,1B,4
B,00,1B,59,29,3E,4C
2490 data 6F,61,64,69,6E,67,20,44,69,6
3,74,69,6F,6E,61,72
2500 data 79,1B,4B,00,1B,59,29,3E,53,6
1,76,69,6E,67,20,44
2510 data 69,63,74,69,6F,6E,61,72,79,1
B,4B,00,1B,59,29,3E
2520 data 44,69,63,74,69,6F,6E,61,72,7
9,20,4E,6F,74,20,46
2530 data 6F,75,6E,64,20,28,52,29,65,7
4,72,79,20,28,43,29
2540 data 72,65,61,74,65,1B,4B,00,1B,5
9,29,3E,44,69,63,74
2550 data 69,6F,6E,61,72,79,20,53,61,7
6,65,20,45,72,72,6F
2560 data 72,20,28,52,29,65,74,72,79,2
0,28,43,29,61,6E,63
2570 data 65,6C,1B,4B,00,1B,59,29,3E,4
4,69,63,74,69,6F,6E
2580 data 61,72,79,20,4C,6F,61,64,20,4
5,72,72,6F,72,20,28
2590 data 43,29,61,6E,63,65,6C,1B,4B,0
0,1B,59,29,3E,49,6E
2600 data 70,75,74,20,46,69,6C,65,20,4
E,6F,74,20,46,6F,75
2610 data 6E,64,20,28,52,29,65,74,72,7
9,20,28,43,29,61,6E
2620 data 63,65,6C,1B,4B,00,1B,59,29,3
E,4F,75,74,70,75,74
2630 data 20,46,69,6C,65,20,4F,70,65,6
E,20,45,72,72,6F,72
2640 data 20,28,52,29,65,74,72,79,20,2
8,43,29,61,6E,63,65
2650 data 6C,1B,4B,00,1B,59,29,3E,53,6
1,76,65,20,44,69,63
2660 data 74,69,6F,6E,61,72,79,20,55,7
0,64,61,74,65,73,20
2670 data 28,59,29,65,73,20,28,4E,29,6
F,1B,4B,00,1B,59,29
2680 data 3E,54,68,61,6E,6B,73,20,66,6
F,72,20,75,73,69,6E
2690 data 67,20,53,70,65,6C,6C,62,69,6
E,64,65,72,21,1B,4B
2700 data 00,1B,59,2B,3E,3E,20,1B,4B,0
0,1B,59,2B,3E,3E,20
2710 data 50,6C,65,61,73,65,20,77,61,6
9,74,2E,2E,2E,1B,4B
2720 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2730 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2740 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2750 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2760 data 00,00,00,00,1B,70,00,1B,71,0
0,1B,6A,00,1B,6B,00
2770 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2780 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2790 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2800 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2810 data 00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2820 data 00,00,00,00,00,00,00,00,00,0
```

```
0,00,00,00,00,00,00
2830 data 00,00,00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2840 data 00,00,00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,00
2850 data 00,00,00,00,00,00,00,00,3E,1A,0
A,14,10,06,06,0C,0E
2860 data 14,0A,1E,06,0A,0A,06,06,18,0
A,0A,14,06,06,06,10
2870 data 0E,14,0A,20,08,0E,14,0A,20,0
8,0C,1C,26,0C,14,12
2880 data 10,3A,0E,04,0E,0E,10,06,06,3
C,0A,0A,0A,0E,0A,0A
2890 data 0A,08,24,34,0C,08,0C,0A,14,0
6,06,0C,1E,0A,08,0C
2900 data 06,0E,0E,0C,0A,0A,1C,0E,14,0
A,1E,06,0A,0A,06,06
2910 data 12,08,12,0A,14,0C,06,12,00
2920 data *
```

●

ST CHECKSUM DATA.

(see page 84)

```
 100 data  68, 948, 117, 614, 503, 253
, 410, 427, 14, 109, 3463
 190 data  685, 357, 602, 636, 895, 74
4, 838, 754, 846, 804, 7161
 1080 data  218, 908, 920, 824, 824, 8
44, 767, 816, 845, 780, 7746
 1180 data  935, 900, 913, 861, 36, 89
5, 958, 826, 854, 849, 8027
 1280 data  829, 31, 759, 911, 831, 79
2, 986, 841, 930, 772, 7682
 1380 data  730, 737, 780, 944, 848, 8
76, 962, 658, 797, 833, 8165
 1480 data  918, 842, 946, 934, 973, 7
00, 790, 858, 975, 907, 8843
 1580 data  927, 863, 837, 904, 875, 8
82, 676, 683, 830, 962, 8439
 1680 data  895, 910, 849, 921, 682, 9
11, 851, 907, 896, 54, 7876
 1780 data  739, 883, 842, 899, 911, 9
25, 832, 840, 918, 880, 8669
 1880 data  866, 44, 895, 910, 802, 81
5, 942, 14, 14, 968, 6270
 1980 data  847, 735, 778, 719, 661, 6
52, 649, 676, 699, 696, 7112
 2080 data  914, 693, 652, 649, 676, 7
76, 690, 762, 717, 727, 7256
 2180 data  764, 720, 773, 715, 724, 7
66, 720, 763, 720, 720, 7385
 2280 data  755, 745, 754, 768, 811, 9
16, 915, 869, 924, 898, 8355
 2380 data  824, 877, 846, 966, 897, 8
73, 937, 896, 888, 897, 8901
 2480 data  917, 955, 882, 927, 931, 8
67, 881, 919, 845, 937, 9061
 2580 data  884, 913, 914, 860, 918, 9
06, 846, 877, 912, 863, 8893
 2680 data  953, 925, 861, 919, 542, 5
43, 544, 545, 744, 547, 7123
 2780 data  548, 549, 543, 544, 545, 5
46, 547, 677, 722, 752, 5973
 2880 data  796, 769, 790, 595, 218, 3
168
```

●

Listing 2.
Assembly listing.

```
        text
************************************
*** Spellbinder Spelling Checker ***
***        by Kirk Stover        ***
************************************

gemdos      equ   $01
term        equ   $00
conin       equ   $01
rawconio    equ   $06
conws       equ   $09
conrs       equ   $0a
create      equ   $3c
open        equ   $3d
close       equ   $3e
read        equ   $3f
write       equ   $40
malloc      equ   $48
setblock    equ   $4a

main        move.l    sp,a5               ; a5 has current stack pointer
            move.l    4(a5),a5            ; base page start
            move.l    $c(a5),d0           ; text segment length
            add.l     $14(a5),d0          ; data segment length
            add.l     $1c(a5),d0          ; bss segment length
            add.l     #$100,d0            ; base page offset
            move.l    d0,-(sp)            ; program length to save
            move.l    a5,-(sp)            ; starting address of program
            move.w    #0,-(sp)            ; dummy parameter
            move.w    #setblock,-(sp)     ; setblock function
            trap      #gemdos             ; call gemdos routine
            add.l     #12,sp              ; restore stack

            bsr       initialize          ; allocate memory
            bsr       load_dict           ; load dictionary file
            bsr       open_in             ; open input file
            bsr       open_out            ; open output file
            lea       clear_scr,a0        ; clear screen for processing
            bsr       write_str           ;
            bsr       proofread           ; read and match
            bsr       save_dict           ; save dictionary
exit        move.w    #term,-(sp)         ; return to the desktop
            trap      #gemdos


initialize  move.b    #0,dict_flag        ; turn off dictionary update flag
            move.l    #0,dict_len         ; initialize dict length to 0
            move.l    #-1,-(sp)           ; return amount of memory
            move.w    #malloc,-(sp)       ;    that is available
            trap      #gemdos             ; call gemdos
            addq.l    #6,sp               ; restore stack
            move.l    d0,max_dict         ;
            move.l    d0,-(sp)            ; request all available memory
            move.w    #malloc,-(sp)       ;
            trap      #gemdos             ; call gemdos
            addq.l    #6,sp               ; restore stack
            move.l    d0,dict_buff        ; pointer to start of free area
            move.l    d0,dict_end         ; dictionary is empty for now
            lea       title_msg,a0        ;
            bsr       write_str           ; print the title screen
            rts                           ;


load_dict   lea       dfile_msg,a0        ; request dictionary name
            bsr       get_name            ;
            move.w    #0,-(sp)            ; open dictionary for
            move.l    #file_name,-(sp)    ;    read only
            move.w    #open,-(sp)         ;
            trap      #gemdos             ; call gemdos
            addq.l    #8,sp               ; restore stack
            tst.w     d0                  ; did an open error occur?
            bpl       load_dict_2         ;    no, go load it
            lea       derri_msg,a0        ;    yes
            bsr       write_str           ;       prompt for retry or create
load_dict_1 lea       prompt_msg,a0       ;
            bsr       write_str           ;
            bsr       read_char           ; get response
            cmp.b     #'r',d0             ; retry?
            beq       load_dict           ;    yes, try again
            cmp.b     #'c',d0             ; create?
            bne       load_dict_1         ;    no, invalid response
            bra       load_dict_4         ;    yes, exit from load
load_dict_2 move.w    d0,dict_handle      ; save dictionary handle
            lea       load_msg,a0         ; print loading
            bsr       write_str           ; and
            lea       wait_msg,a0         ; wait
            bsr       write_str           ; messages
            move.l    dict_buff,-(sp)     ; pass dictionary buffer area
            move.l    max_dict,-(sp)      ;      maximum dict length
            move.w    dict_handle,-(sp)   ;      dictionary handle
            move.w    #read,-(sp)         ;      read option
            trap      #gemdos             ;      call gemdos
            add.l     #12,sp              ; restore stack
            tst.l     d0                  ; did a read error occur?
            bmi       load_dict_2a        ;    yes
            cmp.l     max_dict,d0         ; did we read in maximum size?
            bne       load_dict_3         ;    no, continue processing
```

```
load_dict_2a lea      derr3_msg,a0        ; print load error
             bsr      write_str           ; message
             lea      prompt_msg,a0       ;
             bsr      write_str           ;
             bsr      read_char           ; wait for keypress
             addq.l   #4,sp               ; remove return address from
             bra      exit                ; stack and exit program
load_dict_3  move.l   d0,dict_len         ; store actual file size
             add.l    dict_buff,d0        ; calculate dictionary end
             move.l   d0,dict_end         ; and store result
             move.w   dict_handle,-(sp)   ; close file
             move.w   #close,-(sp)        ;
             trap     #gemdos             ; call gemdos
             addq.l   #4,sp               ; restore stack
load_dict_4  rts                          ;

open_in      lea      ifile_msg,a0        ; request input file name
             bsr      get_name            ;
             move.w   #0,-(sp)            ; open input file
             move.l   #file_name,-(sp)    ; with read mode
             move.w   #open,-(sp)         ;
             trap     #gemdos             ; call gemdos
             addq.l   #8,sp               ; restore stack
             tst.w    d0                  ; did an error occur?
             bpl      open_in_2           ;    no
             lea      ierr_msg,a0         ;    yes, prompt for
             bsr      write_str           ;       retry or cancel
open_in_1    lea      prompt_msg,a0       ;
             bsr      write_str           ;
             bsr      read_char           ; wait for keypress
             cmp.b    #'r',d0             ; retry?
             beq      open_in             ;   yes, try again
             cmp.b    #'c',d0             ; cancel?
             bne      open_in_1           ;   no invalid entry
             addq.l   #4,sp               ;   yes, remove return address
             bra      exit                ;      from stack and exit program
open_in_2    move.w   d0,in_handle        ; save input file handle
             rts                          ;

open_out     lea      ofile_msg,a0        ; request output file name
             bsr      get_name            ;
             move.w   #0,-(sp)            ; create file with
             move.l   #file_name,-(sp)    ; read/write status
             move.w   #create,-(sp)       ;
             trap     #gemdos             ; call gemdos
             addq.l   #8,sp               ; restore stack
             tst.w    d0                  ; did an error occur?
             bpl      open_out_2          ;   no
             lea      oerr_msg,a0         ;   yes, prompt for
             bsr      write_str           ;      retry or cancel
open_out_1   lea      prompt_msg,a0       ;
             bsr      write_str           ;
             bsr      read_char           ; wait for keypress
             cmp.b    #'r',d0             ; retry?
             beq      open_out            ;   yes, try again
             cmp.b    #'c',d0             ; cancel?
             bne      open_out_1          ;   no, invalid response
             addq.l   #4,sp               ;   yes, remove return address
             bra      exit                ;   from stack and exit program
open_out_2   move.w   d0,out_handle       ; save output handle
             rts                          ;

proofread    move.l   #text_buff,-(sp)    ; do priming read of
             move.l   #1,-(sp)            ; input file and
             move.w   in_handle,-(sp)     ; store in text buffer
             move.w   #read,-(sp)         ;
             trap     #gemdos             ; call gemdos
             add.l    #12,sp              ; restore stack
             cmp.l    #1,d0               ; End Of File?
             bne      proofread_6         ;   yes
             move.b   text_buff,d0        ;   no
             cmp.b    #'A',d0             ; check if ascii A-Z or a-z
             blt      proofread_1         ;   no, write to output
             cmp.b    #$5b,d0             ;
             blt      proofread_2         ;   yes, start of word
             cmp.b    #'a',d0             ;
             blt      proofread_1         ;   no, write to output
             cmp.b    #$7b,d0             ;
             blt      proofread_2         ;   yes, start of word
proofread_1  move.l   #text_buff,-(sp)    ; write non alpha ascii
             move.l   #1,-(sp)            ; character to the
             move.w   out_handle,-(sp)    ; output file
             move.w   #write,-(sp)        ;
             trap     #gemdos             ; call gemdos
             add.l    #12,sp              ; restore stack
             clr.w    d0                  ;
             move.b   text_buff,d0        ; place character in low byte
             move.w   d0,-(sp)            ; of word D0
             move.w   #2,-(sp)            ; and write to screen
             trap     #gemdos             ; call gemdos
             addq.l   #4,sp               ; restore stack
             bra      proofread           ; go and try again
proofread_2  move.l   #text_buff,a3       ; start of word
proofread_3  addq.l   #1,a3               ; point to next character
             move.l   a3,-(sp)            ; and read
             move.l   #1,-(sp)            ; another byte
             move.w   in_handle,-(sp)     ; from the input file
             move.w   #read,-(sp)         ;
             trap     #gemdos             ; call gemdos
             add.l    #12,sp              ; restore stack
             cmp.l    #1,d0               ; End OF File?
             bne      proofread_5         ;   yes
             move.b   (a3),d0             ;   no
```

```
                cmp.b   #'A',d0         ; check if ascii A-Z or a-z
                blt     proofread_4     ;   no, end of word
                cmp.b   #$5b,d0         ;
                blt     proofread_3     ;   yes, get next character
                cmp.b   #'a',d0         ;
                blt     proofread_4     ;   no, end of word
                cmp.b   #$7b,d0         ;
                blt     proofread_3     ;   yes, get next character
proofread_4     move.b  d0,save_byte    ; save delimitting byte
                move.b  #0,(a3)         ; mark end of word with 0
                bsr     match_dict      ; try to match with dictionary
                move.b  save_byte,text_buff ; restore delimitting byte
                bra     proofread_1     ; and go print it
proofread_5     bsr     match_dict      ; try to match last word
proofread_6     move.w  in_handle,-(sp) ; close input file
                move.w  #close,-(sp)    ;
                trap    #gemdos         ; call gemdos
                addq.l  #4,sp           ; restore stack
                move.w  out_handle,-(sp) ; close output file
                move.w  #close,-(sp)    ;
                trap    #gemdos         ; call gemdos
                addq.l  #4,sp           ; restore stack
proofread_x     rts                     ;

match_dict      move.l  dict_buff,a3    ;
match_dic_1     lea     text_buff,a2    ; start at beginning of text to match
                cmp.l   dict_end,a3     ; is dictionary at end?
                beq     match_dic_5     ;   yes, no match found on dict
match_dic_2     move.b  (a3)+,d3        ; store match value for dict
                move.b  (a2)+,d2        ; store match value for text
                cmp.b   #13,d3          ; is dict on end of word?
                bne     match_dic_3     ;    no, check text
                cmp.b   #0,d2           ;    is text on end of word?
                beq     match_dic_6     ;       yes, match was found
                addq.l  #1,a3           ;       no, adjust pointer
                bra     match_dic_1     ;           and try next dict entry
match_dic_3     cmp.b   #0,d2           ; is text on end of word?
                beq     match_dic_4     ;   yes, skip to next dict entry
                or.b    #$20,d2         ; make all lower case for match
                cmp.b   d2,d3           ; does text = dict
                beq     match_dic_2     ;    yes, try next character
match_dic_4     move.b  (a3)+,d0        ; move dict pointer until
                cmp.b   #10,d0          ;    it points to the current
                bne     match_dic_4     ;    end of word, then
                bra     match_dic_1     ;    try to match next entry
match_dic_5     lea     save_pos,a0     ; save current cursor position
                bsr     write_str       ;
                lea     rev_on,a0       ; turn reverse video on
                bsr     write_str       ;
                lea     text_buff,a0    ; print highlighted word
                bsr     write_str       ;
                lea     rev_off,a0      ; turn reverse video off
                bsr     write_str       ;
                bsr     correct_it      ; go to options
                lea     load_pos,a0     ; restore cursor position
                bsr     write_str       ;
match_dic_6     lea     text_buff,a0    ; print word on screen
                bsr     write_str       ;
                lea     text_buff,a0    ; determine word length
                bsr     find_len        ;
                move.l  #text_buff,-(sp) ; and write it to the
                move.l  d0,-(sp)        ; output file
                move.w  out_handle,-(sp) ;
                move.w  #write,-(sp)    ;
                trap    #gemdos         ; call gemdos
                add.l   #12,sp          ; restore stack
                rts                     ;
find_len        clr.l   d0              ; intialize word length counter
find_len_1      cmp.b   #0,(a0)+        ; check for delimitting 0
                beq     find_len_2      ;
                addq.l  #1,d0           ; increment counter
                bra     find_len_1      ; try again
find_len_2      rts                     ;

correct_it      lea     nomatch_msg,a0  ; display correction options
                bsr     write_str       ;
                bsr     read_char       ; wait for keypress
                cmp.b   #'a',d0         ; Add to dictionary?
                bne     correct_it1     ;
                bsr     add_to_dict     ; yes
                bra     correct_it3     ;
correct_it1     cmp.b   #'c',d0         ; Change it?
                bne     correct_it2     ;
                bsr     change          ; yes
                bra     correct_it3     ;
correct_it2     cmp.b   #'i',d0         ; Ignore it?
                bne     correct_it      ; invalid response
correct_it3     lea     title_msg2,a0   ; erase correction options
                bsr     write_str       ;
                rts                     ;

add_to_dict     lea     text_buff,a0    ; is there
                bsr     find_len        ; any room left in the
                add.l   dict_len,d0     ; dictionary
                add.l   #2,d0           ;
                cmp.l   max_dict,d0     ;
                blt     add_to_dic1     ; yes, go and update
                lea     full_msg,a0     ; no, display error message
                bsr     write_str       ;   and skip update
                bsr     read_char       ;
                bra     add_to_dic4     ;
add_to_dic1     move.b  #1,dict_flag    ; turn dictionary update flag on
                lea     text_buff,a0    ; point to start of text
                move.l  dict_end,a1     ; point to next dict entry
```

```
add_to_dic2 move.b   (a0)+,d0          }
            add.l    #1,dict_len       } increase dict length
            cmp.b    #0,d0             } end of word?
            beq      add_to_dic3       }   yes
            or.b     #$20,d0           }   no, convert to lower case
            move.b   d0,(a1)+          }      and store in dictionary
            bra      add_to_dic2       }      try next character
add_to_dic3 move.b   #13,(a1)+         } delimit with CR and
            move.b   #10,(a1)+         } LF
            move.l   a1,dict_end       } store new dictionary end
            add.l    #1,dict_len       }
add_to_dic4 rts                        }

change      lea      chg_msg,a0        } display replacement
            bsr      write_str         } prompt
            move.b   #64,text_buff_1   } and read the 64 byte maximum
            move.l   #text_buff_1,-(sp) } string
            move.w   #conrs,-(sp)      }
            trap     #gemdos           } call gemdos
            addq.l   #6,sp             } restore stack
            lea      text_buff,a0      }
            add.w    d0,a0             } point to end of string
            move.b   #0,(a0)           } and delimit with 0
            rts                        }

save_dict   lea      title_msg,a0      } display title screen
            bsr      write_str         }
            cmp.b    #1,dict_flag      } check for updates
            bne      save_dict_4       }   no, skip save
save_dict_1 lea      update_msg,a0     } prompt user
            bsr      write_str         } to save dictionary
            lea      prompt_msg,a0     }
            bsr      write_str         }
            bsr      read_char         } wait for keypress
            cmp.b    #'n',d0           } save?
            beq      save_dict_4       }    no, skip save
            cmp.b    #'y',d0           }
            bne      save_dict_1       }    invalid response
            lea      dfile_msg,a0      } yes, ask for filename
            bsr      get_name          }
            move.w   #0,-(sp)          } create dictionary with
            move.l   #file_name,-(sp)  } read/write mode
            move.w   #create,-(sp)     }
            trap     #gemdos           } call gemdos
            addq.l   #8,sp             } restore stack
            tst.w    d0                } did an open error occur?
            bpl      save_dict_3       }   no, continue
save_dict_2 lea      derr2_msg,a0      }   yes, prompt for retry or cancel
            bsr      write_str         }
            lea      prompt_msg,a0     }
            bsr      write_str         }
            bsr      read_char         } wait for keypress
            cmp.b    #'r',d0           } retry?
            beq      save_dict_1       }   yes, try again
            cmp.b    #'c',d0           } cancel?
            bne      save_dict_2       }   invalid entry
            bra      save_dict_4       }   yes, skip save
save_dict_3 move.w   d0,dict_handle    } save dictionary handle
            lea      save_msg,a0       } print save
            bsr      write_str         } and wait
            lea      wait_msg,a0       } message
            bsr      write_str         }
            move.l   dict_buff,-(sp)   } point to start of buffer
            move.l   dict_len,-(sp)    } write entire length
            move.w   dict_handle,-(sp) }
            move.w   #write,-(sp)      }
            trap     #gemdos           } call gemdos
            add.l    #12,sp            } restore stack
            cmp.l    dict_len,d0       } did it all get written?
            bne      save_dict_2       }   no, go display error
            move.w   dict_handle,-(sp) } close file
            move.w   #close,-(sp)      }
            trap     #gemdos           } call gemdos
            addq.l   #4,sp             } restore stack
            tst.w    d0                } close error?
            bmi      save_dict_2       }   yes, go display error
save_dict_4 lea      done_msg,a0       } display done message
            bsr      write_str         }
            lea      prompt_msg,a0     }
            bsr      write_str         }
            bsr      read_char         } wait for keypress
            rts                        }

get_name    bsr      write_str         } display string pointed to by A0
get_name_1  lea      prompt_msg,a0     }
            bsr      write_str         }
            move.b   #40,file_name_1   } allow for 40 character path name
            move.l   #file_name_1,-(sp) }
            move.w   #conrs,-(sp)      } read string
            trap     #gemdos           } call gemdos
            addq.l   #6,sp             } restore stack
            tst.w    d0                } null string?
            beq      get_name_1        } yes, try again
            lea      file_name,a0      } point to end
            add.w    d0,a0             } of string and
            move.b   #0,(a0)           } delimit with 0
            rts                        }

read_char   move.w   #$ff,-(sp)        } retrieve characters
            move.w   #rawconio,-(sp)   } from keyboard
            trap     #gemdos           } no more are available
            addq.l   #4,sp             } restore stack
            tst.w    d0                }
```

```
            bne     read_char               ;
            move.w  #conin,-(sp)            ; wait for keypress
            trap    #gemdos                 ; call gemdos
            addq.l  #2,sp                   ; restore stack
            or.b    #32,d0                  ; convert to lower case
            rts                             ;

write_str   move.l  a0,-(sp)                ; write string pointed by A0
            move.w  #conws,-(sp)            ; to the screen
            trap    #gemdos                 ; call gemdos
            addq.l  #6,sp                   ; restore stack
            rts                             ;

            data

title_msg   dc.b  27,'E',13,10
            dc.b  '                  /---/|',13,10
            dc.b  '                 /   / |',13,10
            dc.b  '                /   /  |                 ',13,10
            dc.b  27,'p                    ',27,'q',13,10
            dc.b  '              /   /     |',13,10
            dc.b  27,'p Spelling Checker ',27,'q',13,10
            dc.b  '             /   /      |',13,10
            dc.b  27,'p                    ',27,'q',13,10
            dc.b  '           +---+        |',13,10
            dc.b  '           |   |        |',13,10
            dc.b  '           | S |        |',13,10
            dc.b  '           | P |        |',13,10
            dc.b  '           | E |        |',13,10
            dc.b  '           | L |        |',13,10
            dc.b  '           | L |        |',13,10
            dc.b  '           | B |        |',13,10
            dc.b  '           | I |        |',13,10
            dc.b  '           | N |        |',13,10
            dc.b  '           | D |        |',13,10
            dc.b  '           | E |      /',13,10
            dc.b  '           | R |    /',13,10
            dc.b  '           |   |  /',13,10
            dc.b  '           | V |/',13,10
            dc.b  '           |1.0|/',13,10
            dc.b  '           +---+',13,10
            dc.b  0

clear_scr   dc.b  27,'E'
title_msg2  dc.b  27,'H',27,'1',' *** Spellbinder Spelling Checker ***',13,10,0
nomatch_msg dc.b  27,'H',27,'1','No match was found.  Select option '
            dc.b  '(A)dd (C)hange (I)gnore:  ',0
chg_msg     dc.b  27,'H',27,'1','Replace with:  ',0
full_msg    dc.b  27,'H',27,'1','No more room in dictionary!  ',0

dfile_msg   dc.b  27,'Y',41,62,'Enter Dictionary File Name',27,'K',0
ifile_msg   dc.b  27,'Y',41,62,'Enter Input File Name',27,'K',0
ofile_msg   dc.b  27,'Y',41,62,'Enter Output File Name',27,'K',0
load_msg    dc.b  27,'Y',41,62,'Loading Dictionary',27,'K',0
save_msg    dc.b  27,'Y',41,62,'Saving Dictionary',27,'K',0
derr1_msg   dc.b  27,'Y',41,62,'Dictionary Not Found (R)etry (C)reate',27,'K',0
derr2_msg   dc.b  27,'Y',41,62,'Dictionary Save Error (R)etry (C)ancel',27,'K',0
derr3_msg   dc.b  27,'Y',41,62,'Dictionary Load Error (C)ancel',27,'K',0
ierr_msg    dc.b  27,'Y',41,62,'Input File Not Found (R)etry (C)ancel',27,'K',0
oerr_msg    dc.b  27,'Y',41,62,'Output File Open Error (R)etry (C)ancel',27,'K',0
update_msg  dc.b  27,'Y',41,62,'Save Dictionary Updates (Y)es (N)o',27,'K',0
done_msg    dc.b  27,'Y',41,62,'Thanks for using Spellbinder!',27,'K',0
prompt_msg  dc.b  27,'Y',43,62,'> ',27,'K',0
wait_msg    dc.b  27,'Y',43,62,'> Please wait...',27,'K',0

dict_handle ds.w  1
in_handle   ds.w  1
out_handle  ds.w  1

file_name_l ds.w  1
file_name   ds.b  40

max_dict    ds.l  1
save_byte   ds.b  1
dict_flag   ds.b  1
dict_buff   ds.l  1
dict_len    ds.l  1
dict_end    ds.l  1

rev_on      dc.b  27,'p',0
rev_off     dc.b  27,'q',0
save_pos    dc.b  27,'j',0
load_pos    dc.b  27,'k',0

text_buff_l ds.w  1
text_buff   ds.w  64

            end
```

Handy-Dandy Slider Subroutines

GEM slider controls to use in your own programs.

by Tom Hudson

The GEM graphic user interface is a convenient way for the ST user to input information into programs. It not only makes the job easier in individual programs, but also helps to create a similarity between programs—which helps users become more comfortable with new software more quickly.

One of the most convenient features of GEM is the "slider" control, seen on the desktop windows. Sliders are a quick way for the user to set a value without using the keyboard.

Unfortunately, there's no ready-made way to create slider controls in your own programs. However, sliders are easy to create with the GEM Resource Construction Set (RCS), and this article will introduce my **Handy-Dandy Slider Subroutines (HDSS)**, a group of painless slider control routines you can use in your own programs. The **HDSS** routines allow the use of up to ten slider controls per dialog, and, with four changes to the code, can be made to handle even more. I think you'll agree that sliders are a convenient way to enter data, and make your programs easier for the novice user to comprehend and use.

Basic slider knowledge.

As stated earlier, GEM doesn't have any ready-made way to create and process slider controls. However, the procedure for building a slider is extremely easy. Let's look at how a slider is constructed.

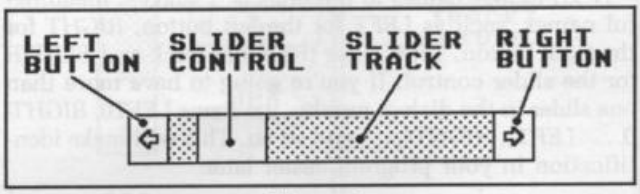


Figure 1.

Figure 1 shows a typical slider control. As you can see, this slider has four parts: the left button, the right button, the slider "track" and the slider control. A vertical slider is constructed in the same way, but the control buttons are termed "up" and "down" (as one would expect).

Each slider is set up to have its range of movement divided into a particular number of steps. For example, a slider that's used to set a number of degrees of rotation will probably be set to range from *0* through *359*. A slider used to set a percentage might run from *0* through *100*.

The slider control moves within the slider track, and to move it, the user points the mouse to the slider control, presses and holds the left mouse button, and drags the slider control to the desired location. The slider control only moves within the track, being limited by the track's borders. When the mouse button is released, the slider is set to the requested position within the track.

The left and right (or up and down) buttons are used to move the slider control 1 unit in the desired direction, and if the mouse button is held down while activating one of these buttons, the 1-unit operation will repeat.

The slider's track is also an active control. If you click on the track to the left of the slider control, the slider control will move one "page" to the left. If you click to the right of the slider control, the control will move one page to the right. The amount of movement that occurs in a track click is programmer defined. More on that later.

Creating the slider.

In order to create a slider quickly and easily, the best way is to use the GEM RCS program. It's possible to set up the structures for a slider without the RCS, but that's beyond the scope of this article.

Once you have your dialog box defined, get two BOX objects and two BOXCHAR objects from the parts selector, and drag them into the dialog. The BOXCHAR objects will be the left and right (or up and down) buttons, and the BOX items will be the slider and slider track.

First, set all four objects so that they are TOUCHEXIT objects. This will cause the dialog box handler to let us know when the user clicks the mouse on any of the objects.

Next, set the characters in the BOXCHAR items so that they have a left arrow or right arrow (or an up or down arrow, if you're making a vertical slider) as their character. These are obtained by the CTRL-A (up arrow), CTRL-B (down arrow), CTRL-C (right arrow), or CTRL-D (left arrow) keystrokes.

Now, select the BOX object that will be used for the track and set it so that it contains a light shading pattern (see Figure 2 for an example). This helps the user identify the track. Leave the slider control an empty white box.

Next, assign names to the objects. I suggest meaningful names, such as *LEFT* for the left button, *RIGHT* for the right button, *TRACK* for the slider track and *SLIDER* for the slider control. If you're going to have more than one slider in the dialog, number the items *LEFT0, RIGHT0, . . . LEFT1, RIGHT1, . . .* and so on. This will make identification in your program easier later.

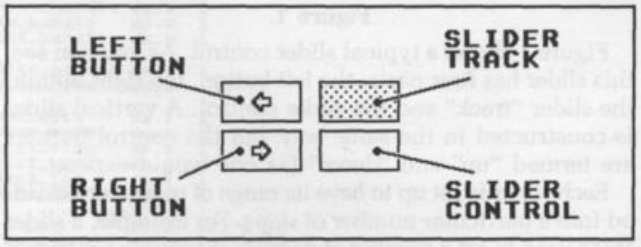


Figure 2.

Now you should have four objects that look something like Figure 2 in your dialog box. We're going to assemble them into a recognizable slider control.

First, stretch the TRACK object to the size and shape you want. You can stretch it vertically for a vertical track, or horizontally for a horizontal track. Place it wherever you like in the dialog box.

Once you've done that, drag the slider control object *on top of* the track object (Figure 3). You'll be warned that the operation will change the structure of the object tree —this is okay. What we're doing is making the slider control a "child" of the track object. This allows us to move the slider control inside the track. To be sure the slider is a part of the track now, move the track to another position. The slider should move with the track. Also, be sure the slider control box is the same width as the slider track box, so that it will slide properly.

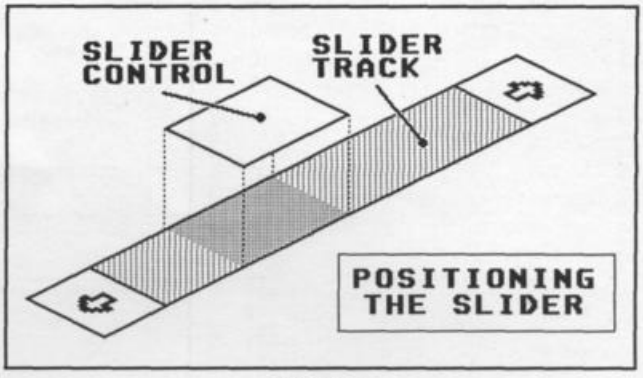


Figure 3.

Finally, position the left and right (or up and down) control objects at either end of the slider track. Now you should have a complete slider control as shown in either Figure 4a or 4b.

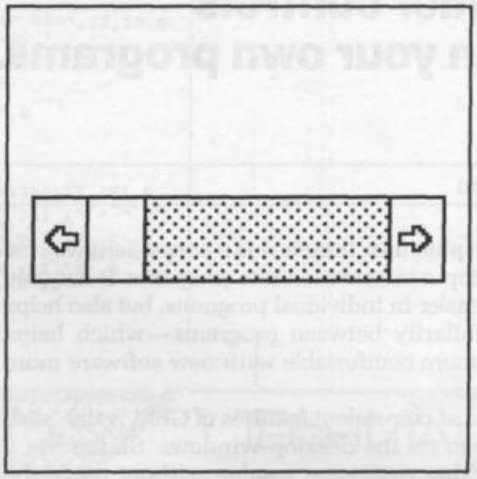


Figure 4a.

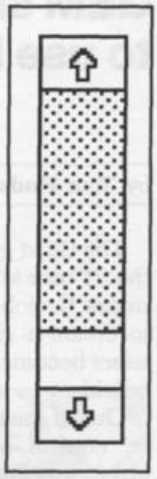


Figure 4b.

Wasn't that easy? Now, we're ready to look at how we'll set up the slider control and check the slider to see if the user has moved it.

The program.

The program that accompanies this article is a simple demonstration of how to control four sliders in one dialog box. Up to ten sliders may be handled at one time with these routines, and you can increase that number to just about any number you like.

The program source is in two parts. Listing 1 is SLIDERS.C, the main program. Listing 2 is SLIDERTN.C, the source code for the **HDSS** subroutines. The output from the RCS, SLIDERS.H, is shown in Listing 3, and Listings 4 and 5 are BASIC loaders that, when run, will create the files SLIDERS.RSC and SLIDERS. DFN (the GEM resource file and RCS definition file), respectively.

To see what the slider example program does, first create the SLIDERS.RSC file by running the ST BASIC program shown in Listing 4. After you've created this file, compile the C listings (see the section "Putting it all together" at the end of this article), and run the resultant .PRG program from the GEM desktop.

Be sure the SLIDERS.RSC resource file is on the same disk. If it isn't, you'll be rudely informed by a GEM alert, and the program will terminate. If you see this alert, just place the SLIDERS.RSC file on the same disk as the program and run it again.

The program starts up by displaying an example dialog with four sliders, two of which are vertical and two horizontal. The dialog is shown in Figure 5, with the slider controls numbered.

each slider control is shown directly beneath the control. Clicking on the various arrow buttons should cause the readout to go up or down one unit.

Now click in the various tracks, on either side of the slider control. You will see the slider control move by the amount indicated in the track increment/decrement value indicated above.

Finally, click on the slider controls themselves and drag the slider controls to different positions. When the mouse button is released, the slider will be set to that position, and the digital readout will reflect the new setting.

After setting the sliders, you can click on OK or CANCEL to exit the dialog. OK will save the slider positions; CANCEL will not. You'll see an alert dialog that will ask you if you'd like to continue or quit. If you click on CON-

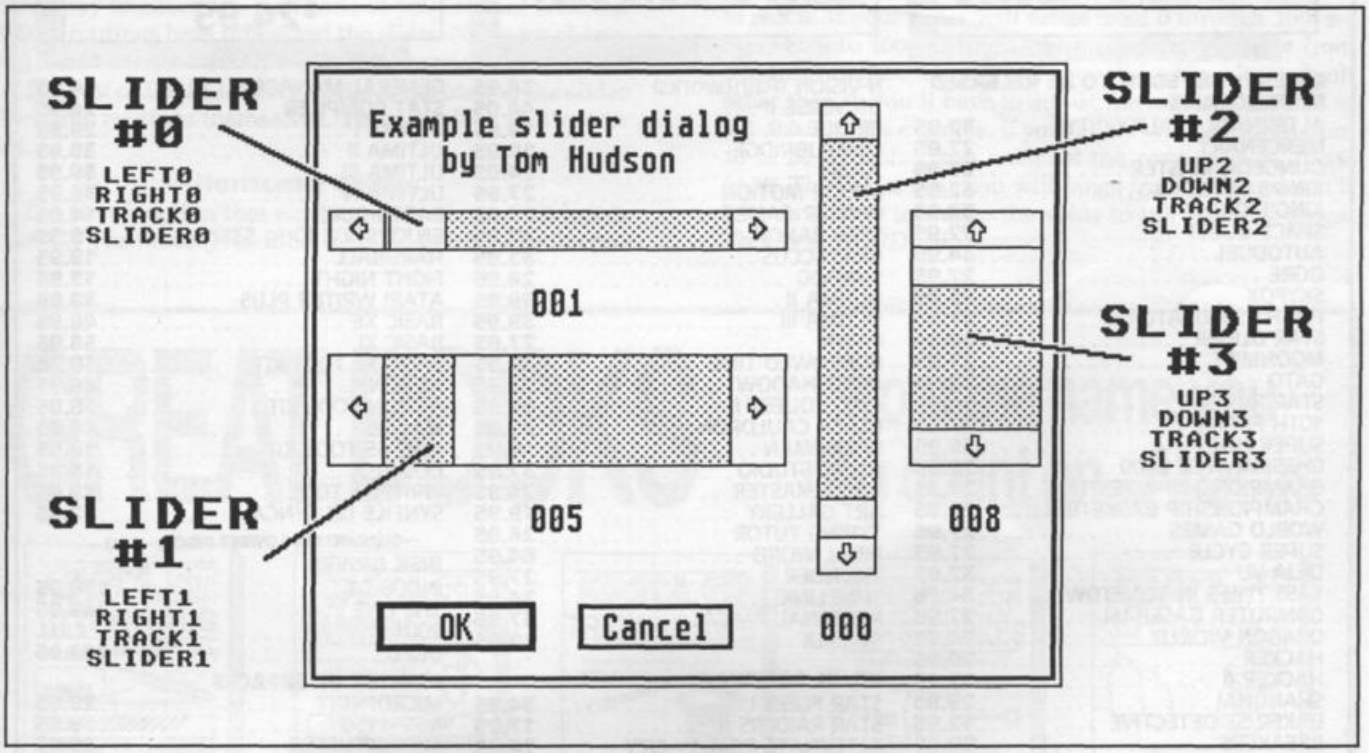


Figure 5.

Slider #0 is set up to range from *0* to *50*. Its initial position is 1. If you click in the slider track (to the left or right of the slider control), the slider will move 5 units in the specified direction. It is a horizontal slider, and its values increase as the slider moves to the right.

Slider #1 ranges from *0* to *20*, with an initial position of *5*. The slider track increment/decrement is 2 units.

Slider #2 ranges from *0* to *100*, with an initial position of *0*. This is a vertical slider, and the values increase as the slider goes toward the top. The track increment/decrement is 10 units.

Slider #3 ranges from *0* to *8*. Its initial position is *8*, and its track increment/decrement is 1 unit.

In all sliders, clicking on the arrow buttons moves the slider one unit in the indicated direction. Try clicking on the arrow buttons and see how they react. The setting of

TINUE, the slider dialog will reappear. You can verify that the slider settings are correct from the last time you set them (if you clicked on OK after setting them). If you click on QUIT, the program will terminate and return to the desktop.

The *HDSS* slider routines.

Take a look at the SLIDERTN.C source file. It contains four C functions and four globally-defined arrays.

The four arrays are for the storage of the parameters for each of the ten sliders. If you have a dialog with more than ten sliders, change the sizes of these four arrays to the appropriate values (be sure all are defined as the same size).

The slidstep[] and slidacc[] arrays are work arrays for **HDSS**. They're used to store the slider movement step value and the accumulated position values, respectively, and you should never alter them. These are LONG variables

COMPUTER GAMES +

(714) 639-8189

520ST or 1040ST CALL

VIP PROFESSIONAL $94.95 (complete version)

130XE or 1050 Drive $129.95

-ST DISK DRIVES-
SF314 d/s $199.95
SF354 s/s $129.95

DB MAN $79.95 (Best ST Data Base)

BOUNTY BOB STRIKES BACK $29.95
RETURN OF HERACLES $24.95

RECENT AND SOON TO BE RELEASED ST PROGRAMS

ALTERNATE REALITY-CITY	32.95
MERCENARY	27.95
DUNGEON MASTER	27.95
KING'S QUEST	32.95
KING'S QUEST III	32.95
SPACE QUEST	32.95
AUTODUEL	34.95
OGRE	27.95
SKYFOX	32.95
FLIGHT SIMULATOR II	37.95
STAR GLIDER	29.95
MOONMIST	27.95
GATO	32.95
STAR RAIDERS	24.95
10TH FRAME	27.95
SUPER HUEY	19.95
CHESSMASTER 2000	29.95
CHAMPIONSHIP WRESTLING	27.95
CHAMPIONSHIP BASKETBALL	29.95
WORLD GAMES	27.95
SUPER CYCLE	27.95
DEJA VU	32.95
TASS TIMES IN TONETOWN	34.95
COMPUTER BASEBALL	27.95
DRAGON WORLD	34.95
HACKER	29.95
HACKER II	32.95
SHANGHAI	29.95
BAKER ST DETECTIVE	12.95
BREAKERS	29.95
VIDEO VEGAS	22.95
RPV	27.95
WORD WRITER ST	59.95
DATA MANAGER ST	59.95
SWIFT CALC ST	59.95
DEGAS ELITE	49.95
LATTICE C	99.95

OTHER OUTSTANDING ST PROGRAMS

SILENT SERVICE	24.95
STAR FLEET I	34.95
HEX	27.95
LEADER BOARD	24.95
TOURNAMENT DISK	14.95
N-VISION (Paintworks)	34.95
UNIVERSE II	44.95
BRIDGE 4.0	19.95
COMPUBRIDGE	19.95
JOUST	24.95
MAJOR MOTION	27.95
WINTER GAMES	27.95
TIME BANDIT	27.95
BRATACCUS	32.95
SUNDOG	24.95
ULTIMA II	39.95
ULTIMA III	39.95
PAWN	27.95
BORROWED TIME	32.95
MINDSHADOW	32.95
KINGS QUEST II	32.95
BLACK CAULDRON	27.95
SPIDERMAN	9.95
MUSIC STUDIO	37.95
PRINT MASTER	24.95
ART GALLERY	19.95
TYPING TUTOR	24.95
FINAL WORD	84.95
THUNDER	27.95
TIME LINK	34.95
PERSONAL PASCAL	47.95
ST TALK	12.95

XE/XL SOFTWARE

STAR FLEET I	34.95
STAR RAIDERS II	17.95
ALTERNATE REALITY-CITY	24.95
ALTERNATE REALITY-DUNGEON	24.95
GUNSLINGER	22.95
221 BAKER ST	22.95
BAKER ST LIBRARY	12.95
NEVER ENDING STORY	22.95
MERCENARY	22.95
VIDEO TITLE SHOP	22.95
THEATRE EUROPE	24.95
LEADER BOARD	27.95
SILENT SERVICE	22.95
F-15 STRIKE EAGLE	22.95
KENNEDY APPROACH	22.95
UNIVERSE	58.95
MICROLEAGUE BASEBALL	24.95
GENERAL MANAGER	24.95
STAT COMPILER	24.95
ULTIMA I	29.95
ULTIMA II	39.95
ULTIMA III	39.95
ULTIMA IV	39.95
PARTY QUIZ	14.95
ENJOYSTICK (Orig $35)	9.95
HARDBALL	19.95
FIGHT NIGHT	19.95
ATARI WRITER PLUS	39.95
BASIC XE	46.95
BASIC XL	36.95
BASIC XL TOOL KIT	18.95
ACTION	46.95
ACTION TOOL KIT	18.95
MAC 65	46.95
MAC 65 TOOL KIT	18.95
DOS XL	18.95
WRITER'S TOOL	38.95
SYNFILE OR SYNCALC	32.95

—GUARANTEED LOWEST PRICES - CALL—

DISK DRIVES

INDUS GT	178.95
THE ONE	249.95
2001	CALL
BIG D	449.95

PRINTER INTERFACES

MICROPRINT	29.95
MPP-1150	39.95
MICROSTUFFER	69.95
UPRINT A-16	69.95
ST PRINTER CABLE	14.95
STAR NX-10 PRINTER	259.95

MODEMS

ATARI XM301	39.95
MPP-1000	34.95
AVATEX 1200	79.95
SUPRA 1200 AT or ST	139.95
ANCHOR VM520	139.95
MPP 300ST	57.95
ST STATION	79.95
ST 20MEG HARD DISK	649.95
ST 1MEG UPGRADE KIT	109.95

TOLL FREE ORDER LINE
Information, Inquiries or
Calif. orders 714-639-8189

1-800-443-8189

MasterCard

SHIPPING: Software - free shipping on U.S. orders over $100, otherwise $2.50 U.S., $6.50 outside U.S. Hardware - depends on weight, call for quote. Charge cards + 3%. C.O.D. orders are welcome, add $1.90 for UPS + 3%.

VISA

COMPUTER GAMES + • Box 6144 • ORANGE CA 92667 • (714) 639-8189

CIRCLE #143 ON READER SERVICE CARD

which are set up to work in a pseudo-fixed-point-math manner. The high word of each holds a whole number value, and the low word holds a fractional value. This gives fractions down to $1/65535$, or .000015, without using floating-point math. The values held in the slidstep[] array are the number of pixels to move per unit of slider movement; slidacc[] is the total offset in pixels for the current position.

The slidpos[] and slidmax[] arrays are the current position and maximum position value for the sliders, respectively. For example, if your slider can range from *0* through *40* and has an initial position value of *24*, the slidmax[] entry for that slider will be *40*, and the slidpos[] entry will be *24*. It's very straightforward.

Out of all these arrays, the only one you need to be concerned with is the slidpos[] array. You read values from this array to check the positions of your sliders after the **HDSS** routines have processed the slider. *Never* try changing these arrays directly—you'll be sorry.

The rest of the SLIDERTN.C file is devoted to the slider handling routines themselves. There are four routines in all.

Horizontal Sliders.

The first function that works with horizontal sliders is hreset(). The hreset() function is called to initialize the position of a horizontal slider, and requires six parameters, as follows:

```
hreset(number,tree,trackix,slidix,
       maximum,initial);
```

Here, number is the number of the slider you would like to initialize, from *0* through *9* (or larger, if you change the four arrays as described above). This is a WORD value.

The tree is the address of the GEM object tree that describes the dialog box. This is a LONG value.

The trackix is the index of the object that is the track portion of the slider. It's a WORD value.

The slidix is the index of the object that is the slider control. This object *must* be a "child" of the object which is that slider's track. This is a WORD value.

The maximum is the maximum value you want the slider to reach. If your slider will range from *0* through *100*, set this value to *100*. The slider always returns a value from *0* to this number, inclusive. If your slider has a lower limit other than *0*, you'll have to adjust the maximum value to a *0-n* range. For example, if your slider will set values from *10* to *20* inclusive, you must set the maximum to *20-10*, or *10*. The values returned will range from *0-10*, and you'll have to add *10* to return the value to the working range. This is a WORD value.

The initial is the setting you want the slider to have when it is drawn, and must range from *0* to the maximum value specified. This is a WORD value.

The hreset() function makes calculations necessary to step the slider control across the track, and stores the values in the slidstep[] and slidacc[] arrays. It then repositions the slider control within the track and redraws the track according to the initial position supplied. Since the slider control is a child of the track object, it is also redrawn. Because the *hreset()* function redraws the track, you must have previously drawn the dialog containing the slider before calling this function.

The second horizontal slider function is the do_hslider() function. When the user clicks the mouse on any of the four objects making up a slider, you call this function, which takes care of all the work involved in monitoring the user action and updating the slider. The function has these parameters:

```
do_hslider(number,tree,trackix,slidix,
  leftix,rightix,whichix,trakstep);
```

In this example, number is the number of the slider, from *0* through *9*. This is a WORD value.

The tree is the address of the GEM object tree that describes the dialog box containing the slider. This is a LONG value.

The trackix is the index of the object that is the slider's track. It is a WORD value.

The slidix is the index of the object that is the slider control. It is a WORD value.

The leftix is the index of the object that is the left movement button on the slider. It is a WORD value. Clicking on this object decreases the setting of the slider.

The rightix is the index of the object that is the right movement button on the slider. It is a WORD value. Clicking on this object increases the setting of the slider.

The whichix is the index of the object that caused the exit from the dialog box. Since all four objects that make up the slider are TOUCHEXIT objects, whenever the user clicks on one of them, the index of that object will be returned by the GEM form_do function. This value is then used as input to the do_hslider() function, so it knows which operation to perform. This is a WORD value.

The trakstep is the number of steps to move in either direction when the mouse is clicked in the track on either side of the slider control. The arrow buttons always move one step in their indicated direction; you can set the trakstep value to any convenient number of steps. This is a WORD value.

When it is called, the do_hslider() function checks to see which of the four slider objects the user selected.

If the mouse was clicked in the slider track, the program determines whether it was on the right or left side of the slider control. It then increments or decrements the slider setting the number of times indicated by the trakstep variable.

If it was the left or right buttons, the function increments or decrements the slider setting, and moves the slider control accordingly. Movement stops at either end of the slider track.

If it was the slider control, the program performs a graf_dragbox() operation, dragging the slider control inside the rectangle defined by the slider track. When the mouse button is released, the graf_dragbox() function returns the slider position, and the program repositions the slider and calculates the position value accordingly.

At the end of the function, the program checks the "moved" work variable to see if the slider was moved. If so, the slider track (and its child, the slider control itself) is redrawn.

That's all the functions for horizontal sliders. The routines are very simple to use, and in a moment, we'll take a look at the workings of the demonstration program to see how they work in practice.

Vertical sliders.

The vertical slider routines, vreset() and do_vslider(), are functionally the same as the horizontal slider functions, except that they work for vertical sliders. Where the values of horizontal sliders increase toward the right side of the slider, the vertical sliders increase toward the top.

The vreset() function works exactly like the hreset() function. It uses the same parameters:

```
vreset(number,tree,trackix,slidix,
       maximum,initial);
```

All the values passed to this function are identical to those in the hreset() function. See the discussion of the hreset() function above for details.

The do_vslider() function is also similar to the do_hslider() function. Its only two differences are in the slider arrow button references. The parameters are:

```
do_vslider(number,tree,trackix,slidix,
    upix,downix,whichix,trakstep);
```

All parameters for this function are identical to those of the do_hslider() function, except two:

The upix is the index of the object that is the up-arrow button on the slider. It is a WORD value. Clicking on this button increases the setting of the slider.

The downix is the index of the object that is the down-arrow button on the slider. It is a WORD value. Clicking on this button decreases the setting of the slider.

This function is identical in operation to the do_hslider() function, except for the fact that it operates with vertically-oriented sliders. Let's look at the demonstration program, which shows how to use the **HDSS** routines.

The demo program.

Now, look at the SLIDERS.C source file. At the start of the file are the #include directives. The most important one here is the <sliders.h> file. This is the output of the RCS program, and contains the names assigned to the various slider object parts and their object numbers. This must be included, so that the program knows what the names of the slider object parts are.

There's only one array that you'll need to have, in order to use the **HDSS** routines. This is the slidpos[10] array, which we define as an external reference (it's defined in the SLIDERTN.C file, and we just need to tell the compiler that it's defined elsewhere).

After setting up the GEM global arrays and the program's working variables, we define the text strings which make up the alert boxes the program will use. These are pretty straightforward, and don't require a great deal of explanation.

The next section of the program defines the variables we will use to store the positions of our four sliders. As you can see, the initial positions we have set up here are *1*, *5*, *0* and *8*, respectively. If you look at Figure 5 again, you'll see that the sliders are indeed set to these values initially. After handling the slider dialog, if the OK button was used, the values in the slidpos[] array will be read and placed in the appropriate slider position variable. If the CANCEL button is used, the positions will not be changed.

In the main section of the program, a virtual workstation is opened and the program attempts to load the GEM resource file, SLIDERS.RSC. If the resource file is not found, an alert dialog is displayed and the program terminates.

If the resource file was properly loaded, the program uses the rsrc_gaddr() function to get the address of the object tree that defines the slider test dialog. It then changes the mouse form to an arrow and makes sure the mouse is displayed on the screen.

The next section of the program uses the form_center(), form_dial() and objc_draw() functions to draw the slider test dialog on the screen. Note that the third parameter of the objc_draw() function is *2*. This is necessary to draw the dialog box (layer 0), the first-level slider objects (layer 1) and the slider control (layer 2). If you make this number lower, not all the slider parts will appear.

Now that the dialog box with the four sliders is drawn, we must tell the **HDSS** routines to initialize all the sliders to their correct positions. This is done by calling the appropriate reset function for each slider. To perform the reset function, we need to furnish the indexes of the tracks and slider controls of each slider. These are found in the SLIDERS.H file, and are shown in Figure 5.

The reset function for slider #0 (a horizontal slider) is:

```
hreset(0,diaaddr,TRACK0,SLIDER0,
       50,slider0);
```

The first parameter, *0*, indicates that we are resetting slider number 0.

The second parameter, *diaaddr*, is the address of the dialog object tree. We got this address from the rsrc_gaddr() function.

The third and fourth parameters are the track and slider control indexes for the first slider. These are found in the SLIDERS.H file.

The fifth parameter, *50*, tells the hreset() function that

the maximum setting of this slider is 50. It can, therefore, range from 0 to 50, inclusive.

The last parameter, slider0, is the initial position of the slider, defined earlier in the program. The first time through, this value will be 1.

The other three sliders are reset in the same way. Remember that horizontally-aligned sliders must be reset with the hreset() function and the vertically-aligned sliders must be reset with the vreset() function.

After resetting the sliders to their initial positions, we use the show_pos() function, defined later in the program, to display the settings of each of the sliders. The three-digit displays are positioned directly beneath the corresponding slider. The function is described in detail below.

The next section of the program starts a loop which first calls the form_do() function, then takes the object number returned by the form_do() and determines whether it was one of the slider objects. If it was, it processes the slider and loops back to continue testing.

The form_do() function will return the value of the object which caused the exit from the dialog. This can either be one of the slider objects (arrow buttons, track or slider control), or the OK or CANCEL buttons. In the case of TOUCHEXIT objects such as the slider controls, a double-click on the objects is indicated by returning the object number with the high-order bit set. This is why the value returned by the form_do() function is logically ANDed with $7FFF—we don't care whether a double-click occurred or not. Once the returned value is stripped of the high-order bit, we have the number of the object that caused the exit from the dialog box.

Now that we have the number of the object the user indicated, we perform a series of tests on the number.

If the object was any of the four objects making up slider #0, we call the do_hslider() function, passing it the slider number (0), the dialog tree address (*diaaddr*), the track, slider, left and right object indexes, the index of the object clicked on by the user and, finally, the number of steps to move if the click was in the slider track (5). The other three sliders are similarly tested and processed if they were clicked on by the user.

If any of the sliders were changed, they are processed by the appropriate slider processor. Their new settings are displayed by the show_pos() function. Then the program loops back to the form_do() call, where the process repeats.

Note that the slider's track increment is passed in the do_hslider() and do_vslider() calls. As you can see by the code, slider 0's track increment is 5, slider 1's is 2, slider 2's is 10, and slider 3's is 1. You can set the track increment to any value that you feel is appropriate.

If the code falls through the slider tests, it checks to see if the form was exited with the OK button (EXITOK). If so, the user wants to use the slider settings, and the program copies them from the slidpos[] array and places them into the individual slider position variables (slider0, slider1, etc.)

If the form was exited by any object other than the CANCEL button (EXITCAN), the program loops back to handle the form.

Once the form is exited via OK or CANCEL, the program displays an alert asking the user to continue or quit. If CONTINUE is selected, the program redisplays the slider test dialog with the current slider settings, so you can see how the OK and CANCEL options affect them. QUIT terminates the program.

Miscellaneous functions.

There are two other functions used by the program. The first, show_pos(), will display a number from 0 through 999 in a dialog box. To do this, a special object must be built. This is simply a BOX object which has a STRING object as a child. The string is initially defined as 999, which is a four-character string (three digits plus a terminating value of 0). Figure 6 shows how this object is constructed.

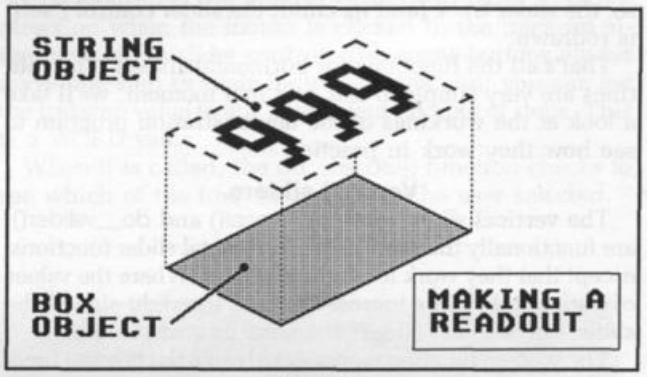


Figure 6.

This construction is necessary for the previous string to be erased (by the box object) and redrawn properly.

To call the show_pos() function, the following parameters are passed:

```
show_pos(tree,index,boxindex,value)
```

Here, tree is the address of the object tree that contains the STRING/BOX object. This is a LONG value.

The index is the object index of the STRING object. It is a WORD value.

The boxindex is the object index of the BOX object containing the STRING object. This is a WORD value.

The value is a WORD containing a value ranging from 0 through 999. This value will be converted into an ASCII string, which is displayed in the STRING object.

This function calls the iset() function. The iset() function converts the value parameter into a string, which is stored in the STRING object, then it redraws the BOX object. Since the BOX object contains the STRING object, the old value is erased (by the box) and replotted.

The iset() function is a very simple integer-to-ASCII conversion routine, which only works on values from 0 through 999. Its parameters are:

```
iset(value,string);
```

Here, value is a WORD containing a number from 0 through 999.

The string is a pointer to the string which is to hold the ASCII output.

Putting it all together.

If you like, you can change some of the settings in the program, such as the slider ranges and track step values, and recompile it to see how your changes work. To do so, compile the SLIDERTN.C and the SLIDERS.C files. Be sure to have the SLIDERS.H file available during the compilation. The result will be two .O files. Then link them, with the AES and VDI libraries, into a program file. If you type in the link command yourself (i.e., if you do *not* use a batch file), the following will work (note that this is *one line* as typed despite the way it's printed here):

```
link68 s.68k=gemstart,sliders,
          slidertn,vdibind,aesbind,
          osbind,gemlib
```

and then:

```
relmod s
```

The result will be the file S.PRG, which you can then run. If you use a batch file to do your linking, then the lines that invoke the linker and RELMOD will look something like the following (again, note that the *link68* invocation is *one line* in the file):

```
link68 %1.68k=gemstart,%1,%2,vdibind,
          aesbind,osbind,gemlib
relmod %1
```

and you would type:

```
link sliders slidertn
```

after clicking on BATCH.TTP, assuming that your batch file was named LINK.

You can use the **HDSS** routines in your own programs, by simply compiling the SLIDERTN.C program to object form and linking it with your programs that need sliders. Just be sure to include the declaration below in your program's source code.

```
extern int slidpos[10];
```

GEM can be a tremendous help when writing complex programs that require a great deal of user interaction. with the **HDSS** in your programs, they will be easier to program—and to use. These routines allow sliders of all sizes and shapes (wide, thin, short, tall, etc.) and should speed your program development considerably. //

---

*Tom Hudson is a free-lance programmer who works primarily with the Atari ST series of microcomputers. His commercial products for the ST include **DEGAS**, **CAD-3D** and **DEGAS Elite**. Tom is a SYSOP in the Atari SIGs on CompuServe, where his ID is 76703,4224.*

Listing 1. — C listing.

```
/**********************************/
/*   GEM Slider Demonstration     */
/*        by Tom Hudson           */
/* for ANALOG Computing Magazine */
/**********************************/

#include <portab.h>
#include <obdefs.h>
#include <osbind.h>
#include <sliders.h>

/**********************************/
/* Slider position table          */
/**********************************/

extern int slidpos[10];
```

Slider Subroutines *continued*

```
/**********************************/
/* GEM's global arrays            */
/**********************************/

int contrl[12];
int intin[128];
int ptsin[128];
int intout[128];
int ptsout[128];

/**********************************/
/* Misc. working variables        */
/**********************************/

int handle,l_intin[11],l_out[57];
int gr_1,gr_2,gr_3,gr_4;
int diax,diay,diaw,diah,formresult;
long diaaddr;

/**********************************/
/* Text strings for alert boxes   */
/**********************************/

char no_rsc[]=
"[1][ | Resource file not on disk! | ][SORRY]";
char again[]=
"[3][ | Continue testing or quit? | ][Continue|Quit]";

/**********************************/
/* Slider position variables      */
/**********************************/

int slider0=1,slider1=5,slider2=0,slider3=8;

/**********************************/
/* Main program starts here       */
/**********************************/

main()
{
register int ix;

appl_init();

handle=graf_handle(&gr_1,&gr_2,&gr_3,&gr_4);

/**********************************/
/* open workstation               */
/**********************************/

for(ix=0; ix<10; ++ix)
 l_intin[ix]=1;
l_intin[10]=2;
v_opnvwk(l_intin,&handle,l_out);

/**********************************/
/* Load resource file             */
/**********************************/

if(rsrc_load("sliders.rsc")==0)
 {
 form_alert(1,no_rsc);
 goto bail_out;
 }

/**********************************/
/* Get dialog address             */
/**********************************/

rsrc_gaddr(0,SLIDIA,&diaaddr);

graf_mouse(0,0L);
v_show_c(handle,0);

/**********************************/
/* Draw dialog and handle user    */
/* interaction with dialog        */
/**********************************/
```

```
redo_form:
form_center(diaaddr,&diax,&diay,&diaw,&diah);
form_dial(0,0,0,0,0,diax,diay,diaw,diah);
objc_draw(diaaddr,0,2,diax,diay,diaw,diah);

/**********************************/
/* Initialize all sliders to      */
/* their current settings         */
/**********************************/

hreset(0,diaaddr,TRACK0,SLIDER0,50,slider0);
hreset(1,diaaddr,TRACK1,SLIDER1,20,slider1);
vreset(2,diaaddr,TRACK2,SLIDER2,100,slider2);
vreset(3,diaaddr,TRACK3,SLIDER3,8,slider3);

/**********************************/
/* Display current settings of    */
/* sliders below each slider      */
/**********************************/

show_pos(diaaddr,NUMBER0,BN0,slidpos[0]);
show_pos(diaaddr,NUMBER1,BN1,slidpos[1]);
show_pos(diaaddr,NUMBER2,BN2,slidpos[2]);
show_pos(diaaddr,NUMBER3,BN3,slidpos[3]);

/**********************************/
/* Do the form!                   */
/**********************************/

formloop:
formresult=form_do(diaaddr,0) & 0x7FFF;

/**********************************/
/* Find which slider to process   */
/* and do it.                     */
/**********************************/

switch(formresult)
 {
 case LEFT0:
 case RIGHT0:
 case SLIDER0:
 case TRACK0:
  do_hslider(0,diaaddr,TRACK0,SLIDER0,LEFT0,RIGHT0,formresult,5);
  show_pos(diaaddr,NUMBER0,BN0,slidpos[0]);
  goto formloop;
  break;
 case LEFT1:
 case RIGHT1:
 case SLIDER1:
 case TRACK1:
  do_hslider(1,diaaddr,TRACK1,SLIDER1,LEFT1,RIGHT1,formresult,2);
  show_pos(diaaddr,NUMBER1,BN1,slidpos[1]);
  goto formloop;
  break;
 case UP2:
 case DOWN2:
 case SLIDER2:
 case TRACK2:
  do_vslider(2,diaaddr,TRACK2,SLIDER2,UP2,DOWN2,formresult,10);
  show_pos(diaaddr,NUMBER2,BN2,slidpos[2]);
  goto formloop;
  break;
 case UP3:
 case DOWN3:
 case SLIDER3:
 case TRACK3:
  do_vslider(3,diaaddr,TRACK3,SLIDER3,UP3,DOWN3,formresult,1);
  show_pos(diaaddr,NUMBER3,BN3,slidpos[3]);
  goto formloop;
  break;
 }

/**********************************/
/* If OK pressed, grab the        */
/* slider settings                */
/**********************************/
```

"IF YOU WANT

THE BEST GRAPHICS PACKAGE FOR THE ATARI ST,

GET DEGAS!" — JACK POWELL, ASSOCIATE EDITOR, ANTIC MAGAZINE

New! D·E·G·A·S Elite

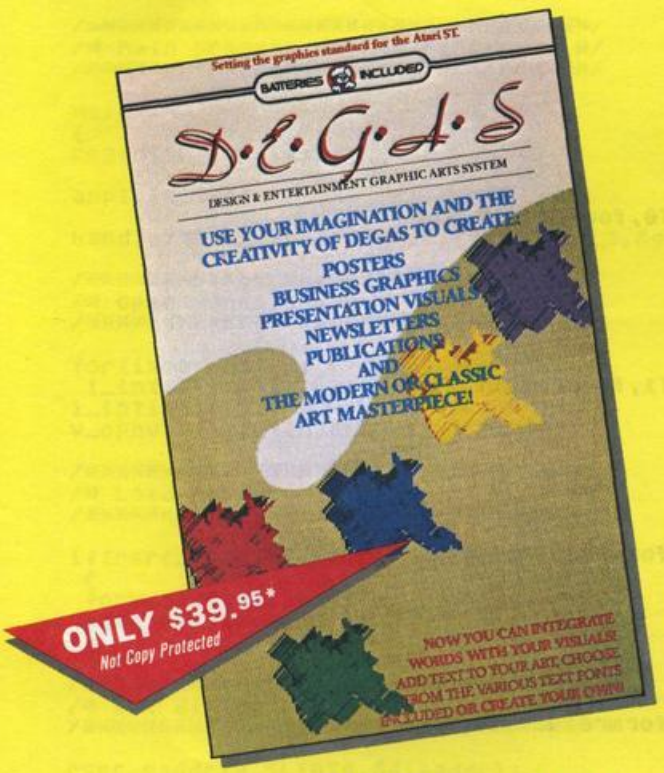
DESIGN & ENTERTAINMENT GRAPHIC ARTS SYSTEM

For Those Who Want Advanced Features

The professional ST graphics package. All the powerful design features that made DEGAS a critical success. Plus, fantastic new capabilities. Now you can create high-quality production artwork for business reports, ads, displays, newsletters and corporate communications.

ADVANCED FEATURES:

- *Up to 8 MULTIPLE WORK SCREENS let you work on several pictures simultaneously*
- *CUT and PASTE between pictures in different screens*
- *BLOCK SAVE entire pictures or portions of pictures to disk — create your own CLIP ART reference files*
- *SCALE, FLIP and ROTATE whole pictures or portions*
- *GRID SNAP aligns your picture elements with expert precision, for ultra-clean layouts*
- *Automatically generates COLOR SHIFT between any two colors*
- *COLOR CYCLING creates movement and virtual animation*
- *Incredible new advanced features including; color animation, block move, text size and font style, color fill patterns, new drawing capabilities, stretch, rotate and much, much, more!*
- *Works with PaperClip Elite word-processor to provide a virtual desktop publishing system*



DEGAS: Design and Entertainment Graphic Arts System

"allows you to take full advantage of your ST's stunning graphics capability, to produce nearly anything you can imagine … created specifically for the ST, sure to become an ST standard."
— ANALOG MAGAZINE

Produce business graphics, posters, newsletters, cartoons — even original works of art. DEGAS makes it easy to exercise your imagination.

DEGAS KEY FEATURES:

- *DRAW by moving the ST mouse — it's your paintbrush, with 15 built-in brush patterns*
- *more than 500 colors to choose from*
- *one command FILLS any outline with a pattern — includes 60 pre-designed patterns, from grids to dots to solid color*
- *total flexibility: create and store your own custom brush patterns, colors, fill patterns and more*
- *powerful TEXT capability lets you combine words and pictures*
- *choose the ideal letter size and weight from the built-in DEGAS TEXT FONT — or design your own font*
- *draw an accurate LINE, CIRCLE, DISK, BOX or FRAME by simply setting two points — DEGAS does the rest automatically*
- *RAYS and K-LINE draw a series of connected lines*
- *MAGNIFY any picture or portion*
- *MOVE or COPY any figure, anywhere on the screen*
- *advanced graphic design tools include MIRROR, SHADOW and AIRBRUSH*
- *one command saves your DEGAS creations to disk*
- *works in all three Atari ST resolution modes*

"Overall Performance: Excellent • Graphics Quality: Excellent • Ease of Use: Easy Value: Excellent • Documentation: Excellent • Error Handling: Excellent •" — FAMILY COMPUTING

DEGAS and DEGAS Elite.

For Atari ST's. Compatible with most popular dot matrix and color printers.

BATTERIES INCLUDED

• BATTERIES INCLUDED, an ITM company, 30 Mural Street, Richmond Hill, Ontario, Canada, L4B 1B5. (416) 881-9941, Customer Information (416) 881-9816
• If you can't find this product at your local retailer, you may order it direct from us at the full suggested list price plus $5.00 for postage and handling. For product orders please call 1-800-387-5707 (U.S. only). • With all Batteries Included products you can have the latest version update of your program by returning the original disk and $10.00.
• Write to us for our full color catalog of products for the APPLE, APPLE MACINTOSH, ATARI, ATARI ST, COMMODORE, COMMODORE AMIGA, and IBM SYSTEMS.

 *ALL PRICES SHOWN ARE IN U.S. DOLLARS. RETAILERS MAY SELL FOR LESS!
You can upgrade the original Degas to the new Degas Elite by returning your disk and $40.00 (U.S.) to Batteries Included.

Integral Solutions

```
if(formresult==EXITOK)
 {
 slider0=slidpos[0];
 slider1=slidpos[1];
 slider2=slidpos[2];
 slider3=slidpos[3];
 }
else

/**********************************/
/* If any button besides CANCEL, */
/* continue form_do.              */
/**********************************/

if(formresult!=EXITCAN)
 goto formloop;

/**********************************/
/* Form exited, reset the button */
/* that caused the exit.         */
/**********************************/

objc_change(diaaddr,formresult,0,diax,diay,diaw,diah,0,1);
form_dial(3,0,0,0,0,diax,diay,diaw,diah);

/**********************************/
/* Ask user whether to continue  */
/* testing sliders or quit.      */
/**********************************/

if(form_alert(1,again)==1)
 goto redo_form;

/**********************************/
/* Free up resource RAM          */
/**********************************/

rsrc_free();

/**********************************/
/* Close the workstation         */
/**********************************/

bail_out:
v_clsvwk(handle);

appl_exit();
}

/*****************************************************/
/* Display slider setting in a three-digit text     */
/* string in the dialog box                         */
/*                                                  */
/* Parameters:                                      */
/* tree     = address of dialog tree                */
/* index    = index of 3-character string object    */
/* boxindex = index of box containing string object */
/* value    = integer number to be displayed (0-999)*/
/*****************************************************/

show_pos(tree,index,boxindex,value)
long tree[][6];
int index,boxindex,value;
{
iset(value,tree[index][3]);
objc_draw(tree,boxindex,1,diax,diay,diaw,diah);
}

/*****************************************************/
/* Convert an integer value (0-999) to a string     */
/*                                                  */
/* Parameters:                                      */
/* num      = integer number to convert (0-999)     */
/* string   = pointer to 4-byte target string       */
/*****************************************************/

iset(num,string)
```

```
int num;
char *string;
{
register int ix,divfac,dct;

for(ix=0,divfac=100; ix<3; ++ix,divfac/=10)
 {
 dct=num/divfac;
 string[ix]=(char)dct+'0';
 num-=(dct*divfac);
 }
string[3]=0;
}
```

●

Listing 2.
C listing.

```
/************************************/
/* Handy-Dandy Slider Subroutines */
/*          by Tom Hudson          */
/* for ANALOG Computing Magazine  */
/************************************/

/*******************************************************/
/* Set up these variables for the slider subroutine */
/* They allow up to 10 sliders per dialog box!      */
/*******************************************************/

long slidstep[10],slidacc[10];
int slidpos[10],slidmax[10];

/*******************************************************/
/* Miscellaneous slider routine variables.  Defined */
/* for use within this module only!                 */
/*******************************************************/

static int mousex,mousey,dum,moved;
static int tbasex,tbasey,sbasex,sbasey;

/*******************************************************/
/* RESET HORIZONTAL SLIDER                          */
/* -----------------------                          */
/* Parameters:                                      */
/* number  = slider number (0-9)                    */
/* tree    = address of dialog tree                 */
/* trackix = index of slider track                  */
/* slidix  = index of slider                        */
/* maximum = maximum slider setting (0-maximum)     */
/* initial = initial slider setting (0-maximum)     */
/*******************************************************/

hreset(number,tree,trackix,slidix,maximum,initial)
int number,tree[][12],trackix,slidix,maximum,initial;
{
slidmax[number]=maximum;
slidpos[number]=initial;
slidstep[number]=((long)(tree[trackix][10]-tree[slidix][10])<<16)/(long)maximum;
slidacc[number]=slidstep[number]*(long)initial;
tree[slidix][8]=(int)(slidacc[number]>>16);
objc_draw(tree,trackix,1,tree[0][8],tree[0][9],tree[0][10],tree[0][11]);
}

/*******************************************************/
/* PROCESS HORIZONTAL SLIDER                        */
/* -------------------------                        */
/* Parameters:                                      */
/* number  = slider number (0-9)                    */
/* tree    = address of dialog tree                 */
/* trackix = index of slider track                  */
/* slidix  = index of slider                        */
/* leftix  = index of left-move button              */
/* rightix = index of right-move button             */
/* whichix = index of item from form_do             */
/* trakstep = # of steps for in-track click         */
/*******************************************************/
```

```
do_hslider(number,tree,trackix,slidix,leftix,rightix,whichix,trakstep)
int tree[][12],trackix,slidix,leftix,rightix,whichix,trakstep;
{
register int ix;
int tempx;

moved=0;

/*******************************************************/
/* If click in track, determine direction of slider */
/* movement & move it TRAKSTEP times in that dir.   */
/*******************************************************/

if(whichix==trackix)
 {
 graf_mkstate(&mousex,&mousey,&dum,&dum);
 objc_offset(tree,slidix,&sbasex,&dum);
 if(mousex>sbasex)
  {
  for(ix=0; ix<trakstep; ++ix)
   {
   if(slidpos[number]<slidmax[number])
    {
    slidacc[number]+=slidstep[number];
    slidpos[number]++;
    moved=1;
    }
   else
    break;
   }
  }
 else
  {
  for(ix=0; ix<trakstep; ++ix)
   {
   if(slidpos[number]>0)
    {
    slidacc[number]-=slidstep[number];
    slidpos[number]--;
    moved=1;
    }
   else
    break;
   }
  }
 }
else

/*******************************************************/
/* Move slider to the right 1 unit.                    */
/*******************************************************/

if(whichix==rightix)
 {
 if(slidpos[number]<slidmax[number])
  {
  slidacc[number]+=slidstep[number];
  slidpos[number]++;
  moved=1;
  }
 }
else

/*******************************************************/
/* Move slider to the left 1 unit.                     */
/*******************************************************/

if(whichix==leftix)
 {
 if(slidpos[number]>0)
  {
  slidacc[number]-=slidstep[number];
  slidpos[number]--;
  moved=1;
  }
 }
else
```

```
/*****************************************************/
/* Clicked on slider -- allow user to drag the       */
/* slider and then record new position               */
/*****************************************************/

if(whichix==slidix)
 {
 /*****************************************/
 /* Get slider screen coordinates         */
 /*****************************************/

 objc_offset(tree,slidix,&sbasex,&sbasey);

 /*****************************************/
 /* Get track screen coordinates          */
 /*****************************************/

 objc_offset(tree,trackix,&tbasex,&tbasey);

 /*****************************************/
 /* Drag the slider inside the track, and */
 /* put final x coordinate in tempx.      */
 /*****************************************/

 graf_dragbox(tree[slidix][10],tree[slidix][11],sbasex,sbasey,
              tbasex,tbasey,tree[trackix][10],tree[trackix][11],
              &tempx,&dum);

 /*****************************************/
 /* If slider was moved, save new position */
 /*****************************************/

 if(tempx!=sbasex)
  {
  slidacc[number]=(long)(tempx-tbasex)<<16;
  slidpos[number]=(int)(slidacc[number]/slidstep[number]);
  slidacc[number]=(long)slidpos[number]*slidstep[number];
  moved=1;
  }
 }

/*****************************************************/
/* Redraw slider track (& slider) if the slider was  */
/* moved by the user's action                        */
/*****************************************************/

if(moved)
 {
 tree[slidix][8]=(int)(slidacc[number]>>16);
 objc_draw(tree,trackix,1,tree[0][8],tree[0][9],tree[0][10],tree[0][11]);
 }
}

/*****************************************************/
/* RESET VERTICAL SLIDER                             */
/* ---------------------                             */
/* Parameters:                                       */
/* number  = slider number (0-9)                     */
/* tree    = address of dialog tree                  */
/* trackix = index of slider track                   */
/* slidix  = index of slider                         */
/* maximum = maximum slider setting (0-maximum)      */
/* initial = initial slider setting (0-maximum)      */
/*****************************************************/

vreset(number,tree,trackix,slidix,maximum,initial)
int number,tree[][12],trackix,slidix,maximum,initial;
{
long work1;

slidmax[number]=maximum;
slidpos[number]=initial;
work1=(long)(tree[trackix][11]-tree[slidix][11])<<16;
slidstep[number]=work1/(long)maximum;
slidacc[number]=work1-slidstep[number]*(long)initial;
tree[slidix][9]=(int)(slidacc[number]>>16);
objc_draw(tree,trackix,1,tree[0][8],tree[0][9],tree[0][10],tree[0][11]);
}
```

```
/*******************************************************/
/* PROCESS VERTICAL SLIDER                             */
/* -----------------------                             */
/* Parameters:                                         */
/* number  = slider number (0-9)                       */
/* tree    = address of dialog tree                    */
/* trackix = index of slider track                     */
/* slidix  = index of slider                           */
/* upix    = index of up-move button                   */
/* downix  = index of down-move button                 */
/* whichix = index of item from form_do                */
/* trakstep = # of steps for in-track click            */
/*******************************************************/

do_vslider(number,tree,trackix,slidix,upix,downix,whichix,trakstep)
int tree[][12],trackix,slidix,upix,downix,whichix,trakstep;
{
register int ix;
int tempy;

moved=0;

/*******************************************************/
/* If click in track, determine direction of slider */
/* movement & move it TRAKSTEP times in that dir.   */
/*******************************************************/

if(whichix==trackix)
 {
 graf_mkstate(&mousex,&mousey,&dum,&dum);
 objc_offset(tree,slidix,&dum,&sbasey);
 if(mousey>sbasey)
  {
  for(ix=0; ix<trakstep; ++ix)
   {
   if(slidpos[number]>0)
    {
    slidacc[number]+=slidstep[number];
    slidpos[number]--;
    }
   }
  moved=1;
  }
 else
  {
  for(ix=0; ix<trakstep; ++ix)
   {
   if(slidpos[number]<slidmax[number])
    {
    slidacc[number]-=slidstep[number];
    slidpos[number]++;
    }
   }
  moved=1;
  }
 }
else

/*******************************************************/
/* Move slider down 1 unit.                            */
/*******************************************************/

if(whichix==downix)
 {
 if(slidpos[number]>0)
  {
  slidacc[number]+=slidstep[number];
  slidpos[number]--;
  moved=1;
  }
 }
else

/*******************************************************/
/* Move slider up 1 unit.                              */
/*******************************************************/

if(whichix==upix)
```

```
 {
 if(slidpos[number]<slidmax[number])
  {
  slidacc[number]-=slidstep[number];
  slidpos[number]++;
  moved=1;
  }
 }
else

/*****************************************************/
/* Clicked on slider -- allow user to drag the       */
/* slider and then record new position               */
/*****************************************************/

if(whichix==slidix)
 {

 /*******************************************/
 /* Get slider screen coordinates           */
 /*******************************************/

 objc_offset(tree,slidix,&sbasex,&sbasey);

 /*******************************************/
 /* Get track screen coordinates            */
 /*******************************************/

 objc_offset(tree,trackix,&tbasex,&tbasey);

 /*******************************************/
 /* Drag the slider inside the track, and   */
 /* put final y coordinate in tempy.        */
 /*******************************************/

 graf_dragbox(tree[slidix][10],tree[slidix][11],sbasex,sbasey,
              tbasex,tbasey,tree[trackix][10],tree[trackix][11],
              &dum,&tempy);

 /*******************************************/
 /* If slider was moved, save new position */
 /*******************************************/

 if(tempy!=sbasey)
  {
  slidacc[number]=(long)(tempy-tbasey)<<16;
  slidpos[number]=slidmax[number]-(int)(slidacc[number]/slidstep[number]);
  slidacc[number]=((long)(tree[trackix][11]-tree[slidix][11])<<16)
                  -(long)slidpos[number]*slidstep[number];
  moved=1;
  }
 }

/*****************************************************/
/* Redraw slider track (& slider) if the slider was */
/* moved by the user's action                        */
/*****************************************************/

if(moved)
 {
 tree[slidix][9]=(int)(slidacc[number]>>16);
 objc_draw(tree,trackix,1,tree[0][8],tree[0][9],tree[0][10],tree[0][11]);
 }
}
```

•

Listing 3.
C listing.

```
#define SLIDIA 0        /* TREE */
#define LEFT0 6         /* OBJECT in TREE #0 */
#define SLIDER0 8       /* OBJECT in TREE #0 */
#define RIGHT0 9        /* OBJECT in TREE #0 */
#define EXITOK 25       /* OBJECT in TREE #0 */
#define EXITCAN 26      /* OBJECT in TREE #0 */
#define TRACK0 7        /* OBJECT in TREE #0 */
```

```
#define TRACK1 16          /* OBJECT in TREE #0 */
#define SLIDER1 17         /* OBJECT in TREE #0 */
#define LEFT1 15           /* OBJECT in TREE #0 */
#define RIGHT1 18          /* OBJECT in TREE #0 */
#define UP2 2              /* OBJECT in TREE #0 */
#define TRACK2 4           /* OBJECT in TREE #0 */
#define SLIDER2 5          /* OBJECT in TREE #0 */
#define DOWN2 24           /* OBJECT in TREE #0 */
#define UP3 10             /* OBJECT in TREE #0 */
#define SLIDER3 12         /* OBJECT in TREE #0 */
#define DOWN3 19           /* OBJECT in TREE #0 */
#define TRACK3 11          /* OBJECT in TREE #0 */
#define NUMBER0 14         /* OBJECT in TREE #0 */
#define NUMBER1 21         /* OBJECT in TREE #0 */
#define NUMBER2 28         /* OBJECT in TREE #0 */
#define NUMBER3 23         /* OBJECT in TREE #0 */
#define BN0 13             /* OBJECT in TREE #0 */
#define BN1 20             /* OBJECT in TREE #0 */
#define BN2 27             /* OBJECT in TREE #0 */
#define BN3 22             /* OBJECT in TREE #0 */
```

•

Listing 4.
ST BASIC listing.

```
100 filename$="a:\SLIDER5.RSC"
110 fullw 2:clearw 2:gotoxy 0,0:print
"creating file..."
120 option base 0
125 dim a%(16000):def seg=1:v$=""
130 p=varptr(a%(0)):bptr=p+1
140 for i%=1 to 818
150 read v$:code%=val("&H"+v$)
160 poke p, code%:print ".";
170 p=p+1
180 next
190 bsave filename$,bptr,818
200 print "file written":end
1000 data 00,00,00,76,00,76,00,76,00,7
6,00,00,00,24,00,76
1010 data 00,00,03,2E,00,1D,00,01,00,0
0,00,00,00,00,00,00
1020 data 00,00,03,32,45,78,61,6D,70,6
C,65,20,73,6C,69,64
1030 data 65,72,20,64,69,61,6C,6F,67,0
0,01,00,62,79,20,54
1040 data 6F,6D,20,48,75,64,73,6F,6E,0
0,04,00,03,00,01,00
1050 data 39,39,39,00,04,00,03,00,02,0
0,39,39,39,00,39,39
1060 data 39,00,02,00,20,20,4F,4B,20,2
0,00,43,61,6E,63,65
1070 data 6C,00,39,39,39,00,FF,FF,00,0
1,00,1B,00,14,00,00
1080 data 00,10,00,02,11,00,00,00,00,0
0,00,29,00,11,00,02
1090 data FF,FF,FF,FF,00,1C,00,00,00,0
0,00,00,00,24,00,03
1100 data 00,01,00,15,00,01,00,03,FF,F
F,FF,FF,00,1A,00,40
1110 data 00,00,00,00,00,3A,00,1C,00,0
1,00,03,00,01,00,04
1120 data FF,FF,FF,FF,00,1C,00,00,00,0
0,00,00,00,3C,00,07
1130 data 00,02,00,0D,00,01,00,06,00,0
5,00,05,00,14,00,40
1140 data 00,00,00,FF,11,11,00,1C,00,0
2,00,03,00,0B,00,04
1150 data FF,FF,FF,FF,00,14,00,40,00,0
0,00,FF,11,00,00,00
1160 data 00,05,00,03,00,01,00,07,FF,F
F,FF,FF,00,1A,00,40
1170 data 00,00,00,00,00,4A,00,01,00,0
4,00,03,00,01,00,09
1180 data 00,08,00,08,00,14,00,40,00,0
0,00,FF,11,11,00,04
1190 data 00,04,00,13,00,01,00,07,FF,F
F,FF,FF,00,14,00,40
1200 data 00,00,00,FF,11,70,00,08,00,0
0,00,03,00,01,00,0A
1210 data FF,FF,FF,FF,00,1A,00,40,00,0
0,00,00,00,4C,00,17
1220 data 00,04,00,03,00,01,00,0B,FF,F
F,FF,FF,00,1A,00,40
1230 data 00,00,00,00,00,4E,00,21,00,0
4,00,07,00,01,00,0D
1240 data 00,0C,00,0C,00,14,00,40,00,0
0,00,FF,11,11,00,21
1250 data 00,05,00,07,00,05,00,0B,FF,F
F,FF,FF,00,14,00,40
1260 data 00,00,00,FF,11,00,00,00,00,0
2,00,07,00,01,00,0F
1270 data 00,0E,00,0E,00,14,00,00,00,0
0,00,00,11,00,00,0C
1280 data 00,06,00,04,00,01,00,0D,FF,F
F,FF,FF,00,1C,00,00
1290 data 00,00,00,00,00,50,00,00,00,0
0,00,03,00,01,00,10
1300 data FF,FF,FF,FF,00,1A,00,40,00,0
0,00,00,00,54,00,01
1310 data 00,08,00,03,00,03,00,12,00,1
1,00,11,00,14,00,40
1320 data 00,00,00,FF,11,11,00,04,00,0
8,00,13,00,03,00,10
1330 data FF,FF,FF,FF,00,14,00,40,00,0
0,00,FF,11,00,00,08
1340 data 00,00,00,03,00,03,00,13,FF,F
F,FF,FF,00,1A,00,40
1350 data 00,00,00,00,00,56,00,17,00,0
8,00,03,00,03,00,14
1360 data FF,FF,FF,FF,00,1A,00,40,00,0
0,00,00,00,58,00,21
1370 data 00,0A,00,07,00,01,00,16,00,1
5,00,15,00,14,00,00
1380 data 00,00,00,00,11,00,00,0C,00,0
C,00,04,00,01,00,14
1390 data FF,FF,FF,FF,00,1C,00,00,00,0
0,00,00,00,5A,00,00
1400 data 00,00,00,03,00,01,00,18,00,1
7,00,17,00,14,00,00
1410 data 00,00,00,00,11,00,00,23,00,0
C,00,04,00,01,00,16
1420 data FF,FF,FF,FF,00,1C,00,00,00,0
0,00,00,00,5E,00,00
1430 data 00,00,00,03,00,01,00,19,FF,F
F,FF,FF,00,1A,00,40
1440 data 00,00,00,00,00,62,00,1C,00,0
```

```
D,00,03,00,01,00,1A
1450 data FF,FF,FF,FF,00,1A,00,07,00,0
0,00,00,00,64,00,04
1460 data 00,0F,00,08,00,01,00,1B,FF,F
F,FF,FF,00,1A,00,05
1470 data 00,00,00,00,00,6B,00,0F,00,0
F,00,08,00,01,00,00
1480 data 00,1C,00,1C,00,14,00,00,00,0
0,00,00,11,00,00,1C
1490 data 00,0F,00,04,00,01,00,1B,FF,F
F,FF,FF,00,1C,00,20
1500 data 00,00,00,00,00,72,00,00,00,0
0,00,03,00,01,00,00
1510 data 00,76
1520 data *
```

●

ST CHECKSUM DATA.
(see page 84)

```
 100 data  996, 948, 117, 614, 503, 17
8, 410, 427, 14, 109, 4316
 190 data  610, 357, 662, 572, 833, 81
8, 784, 675, 748, 820, 6879
 1080 data  555, 913, 917, 578, 931, 5
81, 669, 988, 926, 574, 7632
 1180 data  663, 918, 665, 949, 935, 5
98, 687, 934, 653, 602, 7604
 1280 data  936, 554, 930, 573, 648, 0
, 926, 583, 946, 590, 6686
 1380 data  588, 941, 571, 576, 941, 9
32, 619, 939, 964, 625, 7696
 1480 data  614, 966, 559, 576, 204, 2
919
```

●

Listing 5.
ST BASIC listing.

```
100 filename$="a:\SLIDER5.DFN"
110 fullw 2:clearw 2:gotoxy 0,0:print
"creating file..."
120 option base 0
125 dim a%(16000):def seg=i:v$=""
130 p=varptr(a%(0)):bptr=p+1
140 for i%=1 to 380
150 read v$:code%=val("&H"+v$)
160 poke p, code%:print ".";
170 p=p+1
180 next
190 bsave filename$,bptr,380
200 print "file written":end
1000 data 1B,00,00,00,03,00,53,4C,49,4
4,49,41,00,00,00,00
1010 data 06,00,00,01,4C,45,46,54,30,0
0,00,00,00,00,08,00
1020 data 00,01,53,4C,49,44,45,52,30,0
0,00,00,09,00,00,01
1030 data 52,49,47,48,54,30,00,00,00,0
0,19,00,00,01,45,58
1040 data 49,54,4F,4B,00,00,00,00,1A,0
0,00,01,45,58,49,54
1050 data 43,41,4E,00,00,00,07,00,00,0
1,54,52,41,43,4B,30
1060 data 00,00,00,00,10,00,00,01,54,5
2,41,43,4B,31,00,00
1070 data 00,00,11,00,00,01,53,4C,49,4
4,45,52,31,00,00,00
1080 data 0F,00,00,01,4C,45,46,54,31,0
0,00,00,00,00,12,00
1090 data 00,01,52,49,47,48,54,31,00,0
0,00,00,02,00,00,01
1100 data 55,50,32,00,00,00,00,00,00,0
0,04,00,00,01,54,52
1110 data 41,43,4B,32,00,00,00,00,05,0
0,00,01,53,4C,49,44
1120 data 45,52,32,00,00,00,18,00,00,0
1,44,4F,57,4E,32,00
1130 data 00,00,00,00,0A,00,00,01,55,5
0,33,00,00,00,00,00
1140 data 00,00,0C,00,00,01,53,4C,49,4
4,45,52,33,00,00,00
1150 data 13,00,00,01,44,4F,57,4E,33,0
0,00,00,00,00,0B,00
1160 data 00,01,54,52,41,43,4B,33,00,0
0,00,00,0E,00,00,01
1170 data 4E,55,4D,42,45,52,30,00,00,0
0,15,00,00,01,4E,55
1180 data 4D,42,45,52,31,00,00,00,1C,0
0,00,01,4E,55,4D,42
1190 data 45,52,32,00,00,00,17,00,00,0
1,4E,55,4D,42,45,52
1200 data 33,00,00,00,0D,00,00,01,42,4
E,30,00,00,00,00,00
1210 data 00,00,14,00,00,01,42,4E,31,0
0,00,00,00,00,00,00
1220 data 1B,00,00,01,42,4E,32,00,00,0
0,00,00,00,00,16,00
1230 data 00,01,42,4E,33,00,00,00,00,0
0,00,00
1240 data *
```

●

ST CHECKSUM DATA.
(see page 84)

```
 100 data  989, 948, 117, 614, 503, 15
4, 410, 427, 14, 109, 4285
 190 data  586, 357, 661, 627, 659, 67
8, 726, 694, 631, 659, 6278
 1080 data  648, 636, 607, 682, 695, 5
88, 676, 667, 655, 731, 6585
 1180 data  740, 718, 612, 589, 614, 4
02, 197, 3872
```

●

ST-LOG

THE ATARI ST MONTHLY MAGAZINE

READER SERVICE CARD

Please circle an entry for each category.

1. **Through which of the following did you obtain this copy of *ST-Log*?**
   A Users' group
   B Computer store
   C Newsstand
   D Subscription
   E Borrowed
   F Book store
   G School
2. **How would you rate this issue of *ST-Log*?**
   A Excellent
   B Very good
   C Good
   D Fair
   E Poor
   F Very poor
3. **Which of the following would you use to describe this *ST-Log*? (circle all that apply)**
   A Just right
   B Too complex
   C Too simple
   D Useful
   E Useless
   F Fluff
   G Interesting
   H Boring
   I Biased
4. **What would you like to see *ST-Log* cover more extensively? (choose three)**
   A ST BASIC
   B C language
   C Assembly language
   D Pascal
   E Modula-2
   F Reviews
   G Telecommunications
   H Education
   I Home Applications
   J Business Applications
   K Graphics
   L Music
5. **Which of the following do you own?**
   A 520ST
   B 1040ST
   C DS/DD floppy drive
   D Hard disk
   E Monochrome monitor
   F Color monitor
   G 300-baud modem
   H 1200-baud modem
   I Printer
6. **What is your age?**
   A Under 18
   B 18-24
   C 25-34
   D 35-49
   E 50-64
   F Over 65
7. **How many people, other than yourself, read this issue of *ST-Log*?**
   A One
   B Two
   C Three
   D None
8. **Do you own any of the following computers?**
   A Atari 8-bit
   B Apple II (family)
   C IBM (or compatible
   D Macintosh
   E Amiga
   F Commodore 8-bit
   G Radio Shack Co-Co
   H Other _____
9. **What article do you like best in this issue?** _____
10. **What article do you like least in this issue?** _____

101	121	141	161	181
102	122	142	162	182
103	123	143	163	183
104	124	144	164	184
105	125	145	165	185
106	126	146	166	186
107	127	147	167	187
108	128	148	168	188
109	129	149	169	189
110	130	150	170	190
111	131	151	171	191
112	132	152	172	192
113	133	153	173	193
114	134	154	174	194
115	135	155	175	195
116	136	156	176	196
117	137	157	177	197
118	138	158	178	198
119	139	159	179	199
120	140	160	180	200

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

ST-LOG JANUARY 1987 EXPIRES 3-1-87

MAGAZINE SUBSCRIPTIONS

ANALOG ONLY	ST-LOG ONLY	ANALOG and ST-LOG
☐ 1 yr. (12 iss.) $28.00	☐ 1 yr. (12 iss.) $28.00	☐ 1 yr. (12 iss. each) . . . $42.00
☐ 2 yrs. (24 iss.) $52.00	☐ 2 yrs. (24 iss.) $52.00	☐ 2 yrs. (24 iss. each) . . $78.00
☐ 3 yrs. (36 iss.) $76.00	☐ 3 yrs. (36 iss.) $76.00	☐ 3 yrs. (36 iss. each) . $114.00
CANADIAN & MEXICAN RATES (First Class)		
☐ 1 yr. (12 iss.) $36.00	☐ 1 yr. (12 iss.) $36.00	☐ 1 yr. (12 iss. each) . . . $54.00
☐ 2 yrs. (24 iss.) $68.00	☐ 2 yrs. (24 iss.) $68.00	☐ 2 yrs. (24 iss. each) . $102.00
☐ 3 yrs. (36 iss.) $99.00	☐ 3 yrs. (36 iss.) $99.00	☐ 3 yrs. (36 iss. each) . $149.00
FOREIGN RATES (Surface)		
☐ 1 yr. (12 iss.) $39.00	☐ 1 yr. (12 iss.) $39.00	☐ 1 yr. (12 iss.) $59.00
☐ 2 yrs. (24 iss.) $72.00	☐ 2 yrs. (24 iss.) $72.00	☐ 2 yrs. (24 iss. each) . $112.00
☐ 3 yrs. (36 iss.) $104.00	☐ 3 yrs. (36 iss.) $104.00	☐ 3 yrs. (36 iss. each) . $165.00

☐ Check # _____ ☐ Bill me.
☐ Money Order # _____
☐ MasterCard ☐ VISA
Exp. date _____
Card No. _____

Name _____
Address _____
City _____ State _____ Zip _____

PLEASE ALLOW 4 to 6 WEEKS FOR DELIVERY OF FIRST ISSUE.

DISK/MAGAZINE SUBSCRIPTIONS

ANALOG DISK	ST-LOG DISK	ANALOG and ST-LOG
☐ ½ yr. (6 iss.) $59.00	☐ ½ yr. (6 iss.) $59.00	☐ ½ yr. (6 iss. each) . . . $95.00
☐ 1 yr. (12 iss.) $105.00	☐ 1 yr. (12 iss.) $105.00	☐ 1 yr. (12 iss. each) . . $159.00
CANADIAN & MEXICAN RATES (First Class)		
☐ ½ yr. (6 iss.) $69.00	☐ ½ yr. (6 iss.) $69.00	☐ ½ yr. (6 iss. each) . . $109.00
☐ 1 yr. (12 iss.) $119.00	☐ 1 yr. (12 iss.) $119.00	☐ 1 yr. (12 iss. each) . . $179.00
FOREIGN RATES (Surface)		
☐ ½ yr. (6 iss.) $74.00	☐ ½ yr. (6 iss.) $74.00	☐ ½ yr. (6 iss. each) . . $114.00
☐ 1 yr. (12 iss.) $126.00	☐ 1 yr. (12 iss.) $126.00	☐ 1 yr. (12 iss.) $189.00

☐ Check # _____ ☐ Bill me.
☐ Money Order # _____
☐ MasterCard ☐ VISA
Exp. date _____
Card No. _____

Name _____
Address _____
City _____ State _____ Zip _____

PLEASE ALLOW 4 to 6 WEEKS FOR DELIVERY OF FIRST ISSUE.

PLACE STAMP HERE.
THE POST OFFICE WILL NOT DELIVER MAIL WITHOUT POSTAGE

THE ATARI ST MONTHLY MAGAZINE
ST-LOG

P.O. Box 2367
Clinton, IA 52735

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 31 WORCESTER, MA

THE ATARI ST MONTHLY MAGAZINE
ST-LOG

P.O. Box 625
Holmes, PA 19043

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 31 WORCESTER, MA

THE ATARI ST MONTHLY MAGAZINE
ST-LOG

P.O. Box 625
Holmes, PA 19043

Dr. Flop E. Disk

BONUS DISK PROGRAM

An ST disk editor that lets you see and manipulate data.

by Kirk Stover

It's a lot of fun to get inside a machine like the 520ST. And, if you have the right tools, you can exert a lot of control over what the computer and its peripherals are doing. The disk drive is a perfect sample.

You already know that you can store all of your data on disks, but **Dr. Flop E. Disk** goes one step further—and puts you *in command* of your disk drive. This utility allows you to view and manipulate your data—something all programmers find useful and, often, indispensable.

Written entirely in machine language, the program runs under TOS, accessing GEMDOS and extended BIOS routines. It may sound complicated, but it's very easy to use.

In fact, it performs some tasks automatically. For instance, it "diagnoses" the disk's drivemap, which holds information about the disk. So the **Doctor** knows whether your disk is single or double sided.

Complete instructions for **Dr. Flop E. Disk** are found on the disk version of this issue. This program will prove to be a powerful tool whenever you want to manipulate any data. You're no longer limited to copying an entire file or disk, but can instead copy only that portion of data you really need—regardless of what file it's in. The **Doctor** also allows you to change the data on the disk, and can help in recovering "lost" files. //

Kirk Stover is a Systems Analyst at an insurance company in Minnesota. He enjoys working on his 520ST in both C and assembly language. His special interest is writing time-saving utilities.

ATTENTION PROGRAMMERS!

ST-Log is interested in programs, articles, tutorials and hardware/software review submissions dealing with the ST models. 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 **ST-Log**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ST-Log**. 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.

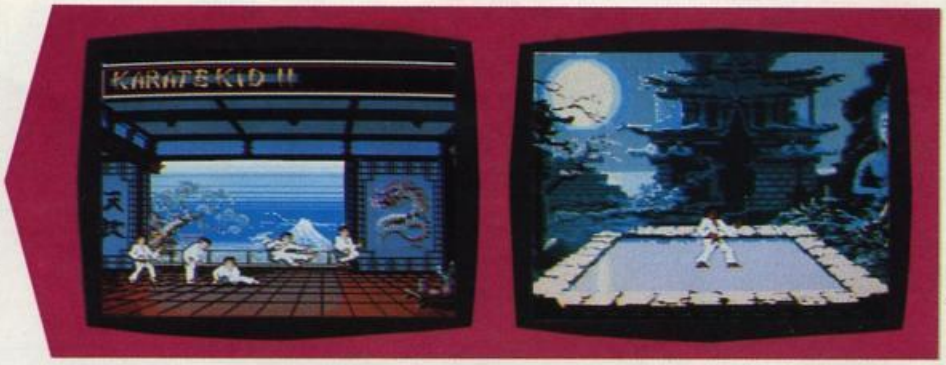
Send programs to:
EDITOR, ST-LOG
P.O. Box 23, Worcester, MA 01603

INDEX TO ADVERTISERS

STIMULATION
FROM THE WORLD'S LARGEST ST PUBLISHER

STupendous!

Karate Kid Part II
Guide Daniel through fight after fight against ever more powerful advesaries. Then face the evil Chozen himself in the Castle of King Shohashi, where you must discover the secret of the drum - or die! But the fighting isn't all: Catch flies with chopsticks and break ice with your bare hand in full-color, animated graphic sequences. Daniel LaRusso faced the challenge and became a hero. What will you do?
Color only. $34.95

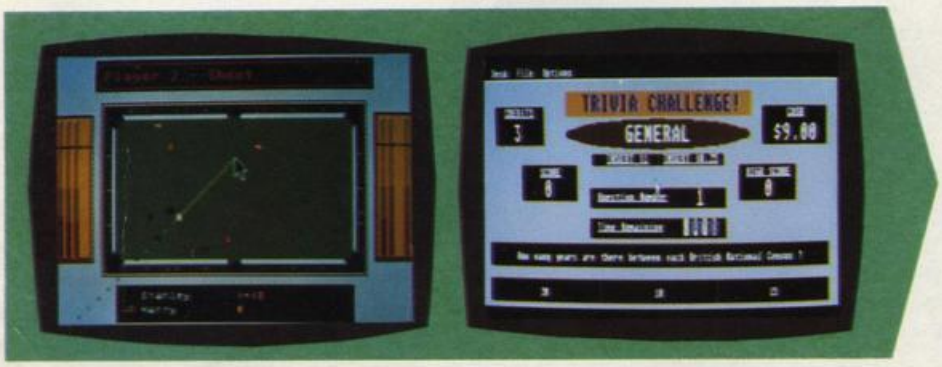


STaggering!

Shuttle II
It's the 1990's and this new Shuttle can take off and fly with the minimum of controls. Decide the launch and landing details then take the seat of Flight Commander for liftoff. Find a faulty Satellite and go out with your jet pack and haul it in. Then return to Earth. Can you keep from burning up? Can you land without crashing? Find out now in this fantastic simulation!
Color only. $34.95

STunning

Pinball Factory
Play great pinball and even design your own screens. Place tabs, bumpers, and ball traps to get the action just right, then use the powerful drawing system to put in the walls. Add flair with a graphic logo. You can even set the scoring, gravity, speed and elasticity! Play the best pinball game you can imagine! There are even tilt controls to let you "bump" the machine!
Color only. $34.95



STriking!

Eight Ball
The poolhall favorite is here! Realistic graphics and ingeniously simple mouse control make this one a sure bet.
Color or Monochrome. $29.95

Trivia Challenge
Nearly 4000 questions on International Sport, The Arts, Pop Music, Science and General Knowledge. Arcade style play keeps the game quick. You can even put in your own questions!
Color or Monochrome. $29.95

Dealer inquiries welcome • Visa and Mastercard accepted • Add $3.00 shipping and handling to each order.

MichTron

576 S. TELEGRAPH, PONTIAC, MI 48053
ORDERS AND INFORMATION (313) 334-5700

CIRCLE #120 ON READER SERVICE CARD

THE STARS

THUNDER *New!*

The writer's assistant

"extremely fast and flexible . . . destined to make a lot of noise!" — COMPUTES ST

3 PROGRAMS IN ONE!

① SPELLING CHECKER 50,000 words
- Real-time mode: program alerts you when you've made a mistake
- Stand-alone mode: have Thunder! check the text after you've finished writing
- Select the correct spelling from a list of similar words — Thunder makes the change instantly

② ABBREVIATION EXPANDER saves you thousands of repetitive keystrokes
- Simply enter a pre-set abbreviation and Thunder! expands it to full form

③ DOCUMENT ANALYZER gives you a statistical report including number of words, average sentence length, FOG score and much more!

ONLY $49.95 * NOT COPY PROTECTED

TIMELINK

Electronic calendar/diary system

"you'll find Timelink indispensable!" — ST WORLD

Your essential tool for planning and record keeping!
- Windows for each day, week, month and year — put any combination on the screen simultaneously
- Instant and automatic updating between frames
- Built-in database capabilities let you keep track of expenses, billable hours and other timely data
- Add/Subtract, Time Between Dates and other features help plan ahead

ONLY $49.95 * NOT COPY PROTECTED

DEGAS

Design and entertainment graphic arts system

"If you want the best graphic arts package for the Atari ST, get DEGAS!" — ANTIC MAGAZINE
- total flexibility: create and store your own custom brush patterns, colors, fill patterns and more
- powerful TEXT capability lets you combine words and pictures
- choose the ideal letter size and weight from the built-in DEGAS TEXT FONT — or design your own font
- draw an accurate LINE, CIRCLE, DISK, BOX or FRAME by simply setting two points — DEGAS does the rest automatically
- MAGNIFY any picture or portion
- MOVE or COPY any figure, anywhere on the screen
- advanced graphic design tools include MIRROR, SHADOW, and AIRBRUSH
- works in all three Atari ST resolution modes

ONLY $39.95 * NOT COPY PROTECTED

ISGUR PORTFOLIO SYSTEM *New!*

Investment manager

"It is dangerous to look at any software package and say, 'This is the one', but this one comes very close." — COMPUTERIZED INVESTING

Helps you manage any combination of investment portfolios more efficiently — stocks, bonds, options, commodities, mutual funds. For casual investors and professionals.
- Designed by Lee Isgur, top Wall Street analyst and V.P. of PaineWebber NY.
- Program updates your portfolios automatically with the latest market data — pre-set for all the popular on-line financial services
- Powerful features show you the most profitable decisions: "Suggest", "Raise Money", "Gains/Losses" and many more
- Analyze, compile and display your investment data in new ways, to gain new insights

ONLY $199.95 * NOT COPY PROTECTED

I*S TALK *New!*

Powerful, easy telecommunications

". . . complete telecommunications that offers you three levels of user definable macros and a 50,000 word memory-resident spelling checker." — PERSONAL COMPUTING

Takes the hassle and uncertainty out of telecommunicating
- Multitude of exceptional features are clearly organized on the GEM user interface
- Call up a HELP window for every major function
- Totally flexible data routing to any combination of devices — disk to modem, modem to memory, screen to modem . . .
- 50,000-word memory-resident, real-time spelling checker to ensure error-free outgoing messages

ONLY $49.95 * NOT COPY PROTECTED

DEGAS ELITE *New!*

For those who want advanced features

The professional ST graphic package. All the powerful design features that made DEGAS a critical success. Plus, fantastic new capabilities.

"Degas Elite is state of the art software!" — ANALOG

ADVANCED FEATURES:
- Up to 8 MULTIPLE WORK SCREENS let you work on several pictures simultaneously
- CUT and PASTE between pictures in different screens
- SCALE, FLIP and ROTATE whole pictures or portions
- Incredible new advanced features including; color animation, block move, text size and font style, color fill patterns, new drawing capabilities, stretch, rotate and much, much more!

ONLY $79.95 * NOT COPY PROTECTED

COMING SOON • PAPERCLIP ELITE!

BATTERIES INCLUDED

BATTERIES INCLUDED, an ITM company, 30 Mural Street, Richmond Hill, Ontario, Canada, L4B 1B5 (416) 881-9941, Customer Information (416) 881-9816

If you can't find this product at your local retailer, you may order it direct from us at the full suggested list price plus $5.00 for postage and handling. For product orders please call 1-800-387-5707 (U.S. only). With all Batteries Included products you can always have the latest version of your program by returning the original disk and $10.00. Write to us for our full color catalog of products for the APPLE, APPLE MACINTOSH, ATARI, ATARI ST, COMMODORE, COMMODORE AMIGA, AND IBM SYSTEMS.

*All prices shown in U.S. funds. Retailers may sell for less.

If you can't find this product at your local retailer call our toll-free number to order: *1-800-387-5707*

MARK WILLIAMS C. AN ENLIGHTENING DEVELOPMENT FOR ATARI ST USERS.

If you've tried your hand at developing applications on the Atari ST, you know the problem. Programming tools aren't only hard to come by, they're hard to use. One might even say primitive. But now for some enlightening news: you can have all the power, portability and versatility of the C language from a leader in professional C programming tools, Mark Williams.

BRING YOUR PROGRAMMING UP TO SPEED.

The Mark Williams C compiler produces fast, dense code and supports the complete Kernighan & Ritchie industry standard C. You'll have access to GEM's AES and VDI libraries for programs using graphics, icons and the Atari mouse. And Mark Williams C lets you take advantage of the full 16 megabytes in Atari's 68000 microprocessor.

STREAMLINE DEVELOPMENT WITH POWER UTILITIES.

Mark Williams C is loaded with everything you'll need for professional development. Bring the power of the UNIX environment to your Atari ST with our Microshell Command Processor including pipes, I/O redirection and more. Edit your program with the highly acclaimed MicroEMACS full screen editor. Accelerate and simplify compiling with *make* which finds and recompiles only those modules affected by your changes. Then, when you're ready for debugging, call on our db Symbolic Debugger with single step, breakpoint and stack traceback functions. Over 40 commands, including a linker and assembler, provide a total development package for your Atari ST.

Features

C compiler
- Complete Kernighan & Ritchie C plus extensions
- Up to eight register variables
- Full access to AES and VDI libraries for programs using graphics, icons and mouse
- Complete UNIX-compatible libraries allow easy portability to and from UNIX development environment.
- Over 300 Atari-specific routines
- One-step compiling, linking with cc command
- English error messages
- Lint-like error checking

Microshell Command Processor, powerful UNIX style shell includes I/O redirection, pipes, command substitutions

MicroEMACS Full Screen Editor with commented source code included

Make Program Building Discipline

Complete symbolic debugger with single-step, breakpoints and stack traceback

Assembler, linker and archiver

Powerful Utilities Package: egrep, sort, diff, cmp, pr, tail, uniq, wc and more

Over 600 pages of documentation including 120 sample C programs

Not copy protected

MARK WILLIAMS C FOR THE ATARI ST
$179.95
60 DAY MONEY BACK GUARANTEE

DEPEND ON A NAME WITH A HISTORY OF PERFORMANCE.

Mark Williams C for the Atari ST is part of our growing line of C compilers. A line that includes the C compiler chosen by DEC, Intel, Wang and thousands of professional programmers. Now our Atari C compiler is earning its own reputation:

"Finally a great C compiler that exploits the power of the ST"–Sigmund Hartmann, President, Atari Software Group

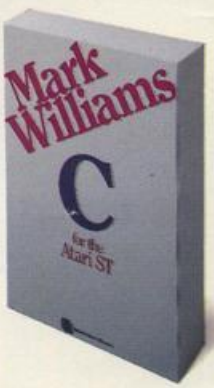
"The all-around best choice for serious software development on the ST."–Douglas Weir of *ANALOG COMPUTING*

GET WHAT YOUR ATARI ST HAS BEEN WAITING FOR.

Mark Williams C is just what your Atari ST was made for: powerful, professional programming. So now that you can have Mark Williams C for just $179.95, what are you waiting for?

Ask your Atari dealer about Mark Williams C or order today by calling **1-800-MWC-1700.***

*In Illinois call: 312-472-6659



Mark Williams Company

1430 West Wrightwood, Chicago, Illinois 60614

CIRCLE #101 ON READER SERVICE CARD