SIX PRINTERS REVIEWED!

ST LOG

THE ATARI ST MONTHLY MAGAZINE

FDC 50076

MAY 1989 ISSUE 31

U.S.A. $3.95
CANADA $4.95

A look at the European ST market

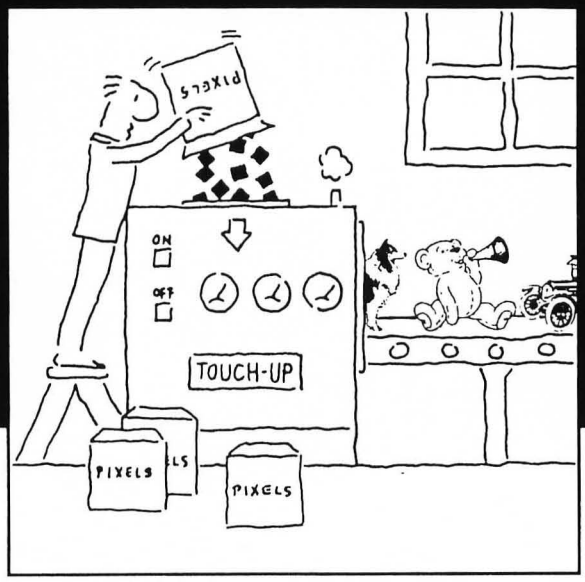
Outline Plus
Line Attack
CES Game Report

Reviews:

CALAMUS
CORRUPTION
ARKANOID
AND MORE!

Give your desktop graphics that professional touch.

Introducing Migraph Touch·Up™



Migraph Touch-Up is the complete design tool for high-resolution monochrome images.

Create, edit, enhance—you can do it all with Touch-Up.

And do it better, because Touch-Up is the first "virtual page" graphics program for the Atari ST.® A sizeable advantage indeed!

Touch-Up can handle bit-mapped images of any size and resolution (based on available memory in your computer). Which means you can now produce pixel-perfect images for all your publishing projects.

Migraph Touch-Up. A powerful tool for professional-quality results.

Ask your dealer for a demonstration of Touch-Up, or call Migraph's toll-free number for more details.

MIGRAPH™ INC.

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

CIRCLE #101 ON READER SERVICE CARD.

EDITORIAL

by Clayton Walnum

Even though Atari has promised to expend more effort on the U.S. market, the ST's main stomping ground still lies on the other side of the Atlantic. In England and Germany particularly, the ST enjoys the kind of success that U.S. owners only dream about.

One way a system's success can be judged is by the amount of third-party products available for it, including software, hardware and even magazines. Software for the ST in Europe is as abundant as snowflakes in the Arctic. Anyone who has managed to find a copy of a European ST magazine has undoubtedly been dumbstruck by the many colorful advertisements for dozens of products, many of which are unheard of in the States.

Luckily, some of these products have managed to trickle into the U.S., some being imported "as is" by ambitious Atari dealers, others being purchased by U.S. software companies and revamped for the American market. These days very little is being developed for the ST in the United States, not only because the market is so small, but because it is much cheaper to bring in already completed products rather than spend a lot of money in research and development. More and more of the ST software we buy is being developed overseas.

Examples? How about *Art & Film Director*, now marketed by Epyx but originally developed in Hungary. How about the many games from Rainbird? All from England. GFA BASIC? From Germany. MichTron's "Microdeal" games? English. And the list goes on.

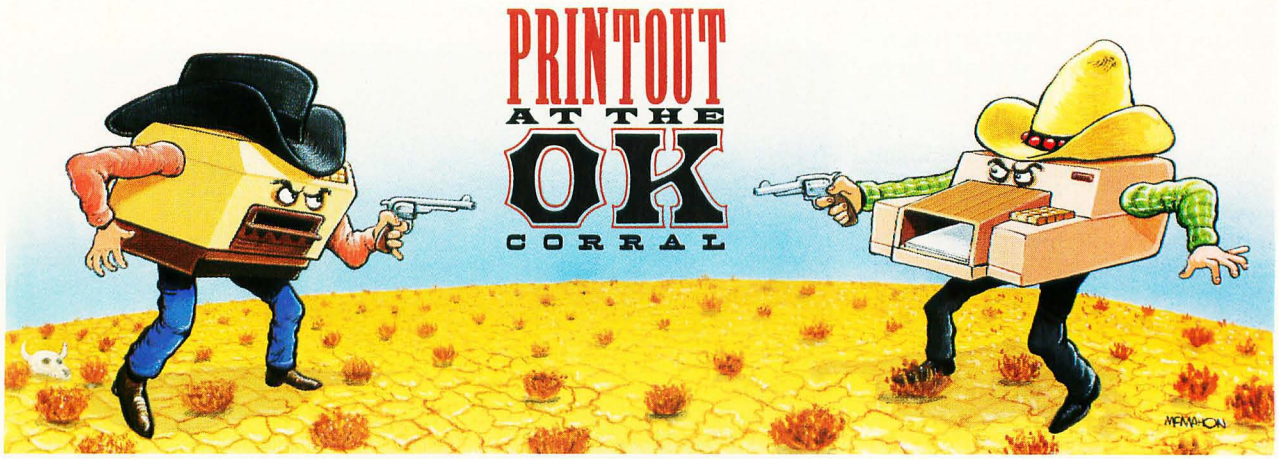
Because we at ST-LOG feel that the European ST market is important, controlling much of what we see in the U.S., we've decided to start a new monthly column, "From Over the Big Water," that will keep you up to date on the ST in Europe. Its author, Marshal M. Rosenthal, is a freelance writer and photographer who has had his ear tuned to the European ST market for a long time. His first installment can be found on page 56.

Also in this issue, you'll find Gregg Anderson's "Printout at the OK Corral," an overview of several of the top printers for your ST. If you've been contemplating a new printer purchase, now's your chance to get the information you need to make a well-informed choice. For students and other writers, James Maki gives us his helpful program *Outline Plus*, an outline generator that will help you organize your writing by first organizing your thoughts. And gamers will get a kick out of Kirk Stover's *Line Attack*. This two-player game is sure to fill your gaming hours with tense action.

In summary, this month, as usual, we have a little something for everybody. Whether you use your ST for desktop publishing, developing animation, programming, gaming or word processing, you'll find articles and reviews that'll help you make the best of your computing time. ▪

IN THIS ISSUE

FEATURES



REVIEWS

ASSEMBLY LINE 66

COLUMNS

DEPARTMENTS

PROGRAM · LISTING · GUIDE

ST-LOG (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. U.S. subscription $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscriptions: add $10 per subscription per year. Single copy $3.50 (add $1 for postage). Change of address: six weeks advance notice, and both old and new addresses are needed. POSTMASTER, send change of address to: **ST-LOG** Magazine, P.O. Box 16928, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

ST LOG

MAY 1989
ISSUE 31

STAFF

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

ADVERTISING SALES

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

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

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

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

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

PERMISSIONS

COVER PHOTOGRAPHY:
LADI VON JANSKY

INTERIOR PHOTOGRAPHY:
HANS WENDLER

ILLUSTRATORS:
JEAN-FRANCOIS PODEVIN
A.W.D.

SUBSCRIPTIONS

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

AUTHORS

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

PD Parade

by George L. Smyth

Note: The program described in this column is available on this month's disk version or in the databases of the ST-LOG SIG on DELPHI.

As I was returning to Washington D.C. from Princeton, New Jersey last month, my car decided to expire. I was eventually able to coax it home, but when I checked the engine over, I realized that I needed to buy a new automobile.

I have never purchased a new car, and I wasn't sure how much I could afford. Then I remembered a public domain program, *SAVELOAN*, which would provide the financial answers I needed. (In my case, the answers were bad news. I decided that another roll of duct tape was more suited to my budget than a new car.)

Loaded as an accessory, *SAVELOAN* can be called from the desktop. Offered are 14 financial programs grouped in three categories: deposits, investments and loans. The left side of the screen displays the 14 routines, that are chosen with the mouse. When the choice is made, the right portion of the screen is blanked and the appropriate prompts are offered. The program I used was labeled "Principal" within the "Loans" section.

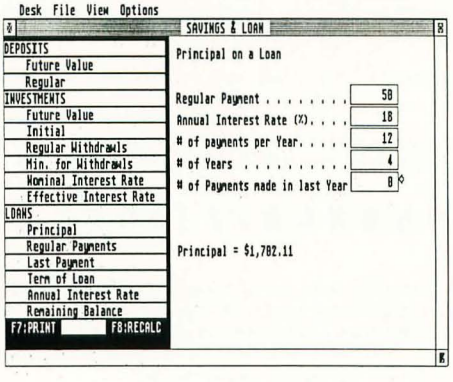
The prompts for this program include, "Regular payment," "Annual interest rate (%)," "numbers of payments per year," "numbers of years" and "numbers of payments made in last year." My input to these were 160, 10, 12, 3 and 0, respectively, assuming the ability to pay $160 per month for three years at an interest rate of 10%. When the last prompt was answered, the calculation from these values was displayed at the bottom: Principal = $4,958.55. (Not much of a car.)

You can modify any of the values using the arrow keys to position the cursor beside the prompt to be altered. The delete or backspace key is used to erase the previous entry, after which the new value can be entered. The two methods of obtaining, in this case, the new principal amount is to either press the Return key until the last prompt is passed or press the F8 function key. F8 recalculates the expression and displays the corrected value in place of the original answer. Changing the number of years from three to four yields a principal of $6,308.45, so that extra year will allow the purchase of a car costing about $1,350 more.

Results can be compared against each other by constantly editing a certain value. For instance, the user may wish to continually change the amount of the payments so that a variety of principals may be obtained. For this reason, sending the entered values and computed sum to a printer becomes useful. (The printer function is performed by pressing the F7 key.) Because the output is straight text there should be no printer compatibility problems.

The routine names are as descriptive as possible, but it's easy to forget which program provides what calculation. Luckily, clicking within a routine's description box will display the associated prompts. This may be done at any time: before, after or while entering values.

The "Deposits" section offers two programs. One routine determines the future value of regular deposits, while the other calculates how much should be regularly deposited to obtain a certain value. It can be an eye-opener to see what a few dollars saved in an orderly manner will yield in just a few years. If paying for college is in your future, this program can give you a real shock.



The "Investments" portion contains routines which determine the future value of an investment, the initial amount needed to be compounded to yield a future value and the amount of regular withdrawals that can be periodically drawn from an initial investment, as well as others. Determining the advantages of differing term and percentage CDs can be done by swapping results back and forth between programs. Since the entered values and results can be sent to the printer for comparison, understanding the tangible differences between all of the bank options is made more palatable.

Most of us will find the "Loans" area to be of greatest interest (no pun intended). Routines to calculate the principal on a loan, the regular payments needed to pay off a principal amount, the annual interest rate on a loan and the remaining balance on a loan will be most often used. When paying off a loan, you don't always have to send the exact amount on the payment stub; you may pay more. It is interesting to use this program to note the amount of money to be saved by increasing payment of a loan by only 10%, considerably less than the total of the remaining stubs.

If you find yourself stuck whenever you try to figure any of the above calculations, it would be difficult to beat *SAVELOAN*. Save this program. You never know: You might be needing a new car soon.

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

YOU NEED THE DISK!

ONLY $9.95 EACH!

If you want to get the most out of ST-LOG, you're also going to want to get your copy of the disk. Each issue's disk contains all the exciting programs for that issue, including the programs whose listings could not be included due to space considerations. The ST-LOG disk version is truly an excellent software value. Order yours today!

MAY 1989	Line Attack, Outline Plus, Monochrome Gray II and more!
MARCH 1989	Picture-Puzzle, Sounds-A-Like, ChemCalc and more!
JANUARY 1989	ST Date OmniLife, Drama-cide, Transwarp and more!
APRIL 1989	Multi-Paint, Font ID Editor, Monochrome Gray and more!
FEBRUARY 1989	Flag Trivia, Super Spool, Desk Switch and more!
DECEMBER 1988	ST Date Planner, Mouse of Fortune, Inside ST Xformer II and more!

YES I DO WANT THE DISK! ONLY $9.95 EACH

- ☐ ST-LOG December 1988 DISK
- ☐ ST-LOG January 1989 DISK
- ☐ ST-LOG February 1989 DISK
- ☐ ST-LOG March 1989 DISK
- ☐ ST-LOG April 1989 DISK
- ☐ ST-LOG May 1989 DISK

Name_____

Address_____

City_____ State_____ Zip_____

☐ Payment Enclosed—Charge My ☐ VISA ☐ MC

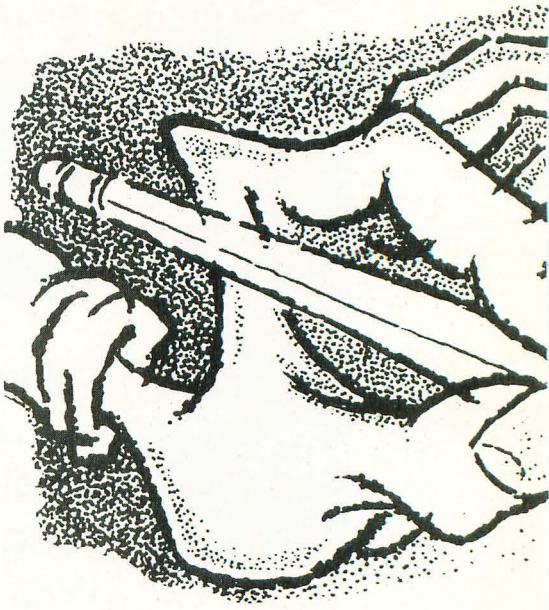
#_____ Exp._____

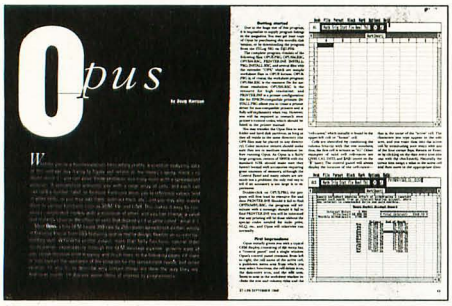
Signature_____

Add $1.50 postage and handling for each disk ordered.
Make check payable to: LFP, Inc. P.O. Box 67068, Los Angeles, CA 90067
California residents add 6.5%

ST LOG
The Atari ST Monthly Magazine

READER COMMENT





Opus Is Best

In my opinion, *Opus* by Doug Harrison (September 1988) is the best program for the Atari ST ever published in ST-LOG. It certainly is also in the top 5% of all programs published for the ST in any ST-specific magazine. However, the main limitation of *Opus* is its "unique" and "proprietary" file structure. So that we users can get the maximum use out of it, could Mr. Harrison or another of the super programmers who write for ST-LOG do a convertor program so that spreadsheets in other popular formats, such as those used by *VIP Professional*, *LDW Power* and *A-Calc Prime*, would be usable with it, as well as be able to port *Opus* data back the other way?

Further, it might not be a bad idea if Mr. Harrison could write an expanded manual explaining in greater detail how to get full use out of *Opus*.—E. Earl Hill
Erie, PA

Thanks for the suggestions. We have been in contact with Doug Harrison about the possibility of doing some follow-up articles to Opus. *Hopefully, you'll find something in* ST-LOG *in the near future.*

Wants Auto-Dial Desk Accessory

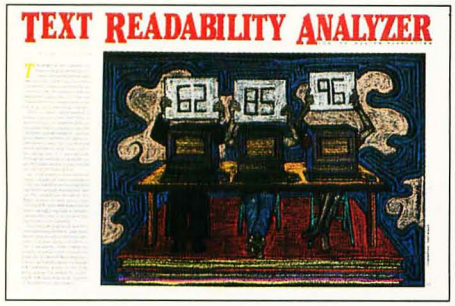
I use *Interlink*, and I would like a desk accessory that will handle a list of bulletin board phone numbers that you would create with *Interlink*'s capture/edit buffer. The accessory would dial the selected number from the list in the background until it got a connection with the bulletin board, and then ring the console bell upon connection.

By doing this in the background, you could execute *Arc Shell* from *Interlink*, unARC the latest *ST-Report*, load it into *Interlink*'s buffer, and be reading while you were waiting to get connected to the next bulletin board. I can program, but multitasking accessories are beyond my capabilities.
—Larry Mears
Huntsville, AL

We're not aware of a program that exactly matches your needs; however, there is one that comes close: DIALXR, *which was published in the July 1988 issue of* ST-LOG. *DIALXR uses the ST's VT-100 emulator to repeat-dial a number while the user works with another application program. The trouble is that it won't help you much with* Interlink. *If there are any programmers out there who have a program that more closely matches what Larry is looking for, feel free to send it along to us for possible use in* ST-LOG.

DIALXR uses the ST's VT-100 emulator to repeat-dial a number while the user works with another application program.

ALL LETTERS TO BE CONSIDERED FOR PUBLICATION SHOULD BE ADDRESSED TO: ST-LOG READER COMMENT P.O. BOX 1413-M.O. MANCHESTER, CT 06040-1413



A Matter of Resolution

I wanted to thank you for the nice presentation of my article, "Text Readability Analyzer," in the February 1989 issue. It might be worth pointing out to the readership, though, that the program runs in high and medium resolution, not low and medium as stated under the title on page 58.

Also, have you ever thought about changing Ian Chadwick's picture to more accurately reflect the mood of the column? Maybe a sneer or a scowl?

—Tom Castle
Richland, MI

Tom is, of course, correct. "Text Readability Analyzer" runs fine on either a color or monochrome monitor. Sorry for the slipup.

As for Ian's sketch, the photo that it was produced from is not really one of Ian, but rather one of Albert Crandy, the winner of the coveted Canadian Home Brewing Award, a prize that's given to the maker of the best home-made beer. Ian substituted that photo for his own because he couldn't find one of himself without a scowl. (This paragraph is a lie.) ■

Porting Between the ST and an 8-bit

I have read the article "Inside the ST XFormer II" (December 1988) and have tried to connect my ST to my 800XL through the modem. I am having difficulty in making the connection. I cannot seem to make the 800XL answer. (I am using *XE-Term.*) How do I go about transferring files from the 8-bit to the ST?

—Cristian Masgras
Columbus, OH

We suspect you're having problems because you're not using a "null modem" cable. A null-modem cable is specially wired for transferring files directly between computers. Your first step in this process, then, is to run down to your local Atari dealer and purchase one of these cables.

Once you have the null-modem cable, it should be connected between the RS-232 ports of your ST and 800XL. You must then run a telecommunications program on each of the machines, making one of them the sender and one the receiver. It's important that you have both telecommunications programs configured the same; that is, they should both be set to the same baud rate, be using the same transfer protocol, etc.

When you've got everything set up as above, all you have to do is send the file to the computer waiting to receive it. We would suggest that, even though you have a direct connection between the computers, that you use a transfer protocol that incorporates error-checking. X-Modem works great. ■

Publisher ST Limitations

I have always relied on magazines such as yours to provide accurate reviews of new products before spending the cold cash to purchase a new program. In the past, your reviews have been very informative and accurate. However, your review of *Timeworks Desktop Publisher ST* (January 1989) didn't mention two major limitations.

Using floppy disks, the program requires multiple disk swaps and is painstakingly slow at times due to the extensive disk access during creation and printing of documents. With a floppy-drive system, your document size is limited by disk space more than memory. Trying to print a two-page document with 12 graphics on a 1040ST with two double-sided drives resulted in an out-of-disk-space error. This was the only document on the disk.

If you have a 9-pin printer, you cannot print the page sideways. The landscape choice prints the page vertically in two sections that must be cut and pasted together. Even then, the right half inch of the page was not printed.

I don't blame you for missing this in your review, since you most likely tried *Publisher ST* on a hard drive and with at least a 24-pin printer. But I do think that your readers should be aware of these limitations.

Keep up the good work.—Fred H. Koch
Colorado Springs, CO

When reviewing a product as complex as Publisher ST, *there are bound to be some details that get missed. We thank you for sharing your observations with us and our readers.* ■

ST NEWS

More MIDI

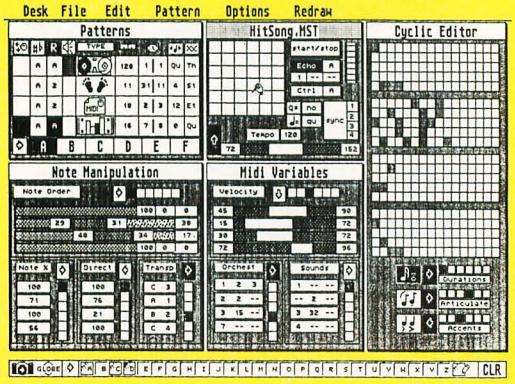
Now available from Intelligent Music is *M*, an interactive composing and performing system. Intelligent Music claims that "*M* is an intelligent musical instrument, a composing and performing program which stimulates your imagination, magnifies your skills and lets you explore ideas in an extraordinary artistic environment."

Music is first developed by assembling the composition's notes and chords. You then define the way the material will be "transformed," after which you perform your music with screen controls and "control keys" on a MIDI keyboard or by using the mouse in a multidirectional grid.

Other features include MIDI-Sync in and out, screen snapshots, patterns with individual loop cycles and independent channelization for each pattern. *M* retails for $200. A demonstration disk is available from Intelligent Music for $5.

Intelligent Music
P.O. Box 8748
Albany, NY 12208
(518) 434-4110

CIRCLE #130 ON READER SERVICE CARD.



New adventure from Sierra

Sierra On-Line, popular publishers of text and graphics adventure games for the ST, have released *Manhunter: New York*, a 3-D animated adventure set in the glorious "Big Apple." *Manhunter* has over 250 different scenes, including accurate maps of New York City, and features split screens, windows, close-ups and a new data storage technique that allows more screens per disk.

According to Sierra, "*Manhunter* is cut from the same mold as Stephen King novels, combining horror and humor. . .when playing *Manhunter* there is more than meets the eye."

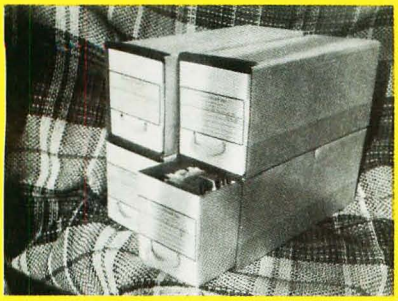
"*Manhunter* takes place in the 'Big Apple' two years after an alien invasion and subsequent world takeover. You will have a chance to play a deadly arcade game in a Flatbush bar, visit a spooky Coney Island carnival and fight off a menacing band of street thugs. Forced into this post-apocalyptic world you must wear the designated brown robe and avoid speaking to other humans, the penalty being death!" *Manhunter* is a huge game, coming on five 3.5-inch disks, and is available now.

Sierra On-Line, Inc.
Coarsegold, CA 93614
(209) 683-4468

CIRCLE #131 ON READER SERVICE CARD.

Low-cost disk storage files

From Weber & Sons comes the new ArchiveDiskFile system #735, each file drawer of which will hold up to 75 3.5-inch disks. The files are constructed of corrugated cardboard and can be used singly or can be stacked and interlocked for greater capacity. According to Weber & Sons, over 675 disks can be stored in one square foot of shelf space.



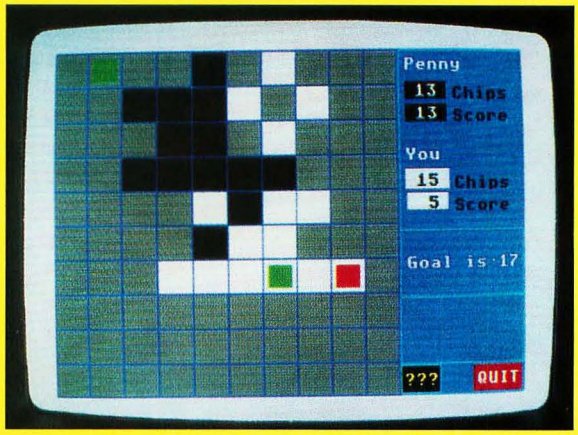
A pack of three files can be purchased for $21 or, for people with extra large storage needs, a 12-pack is available for $78 ($6.50 each).

Weber & Sons, Inc.
3468 Highway 9
Freehold, NJ 07728
(201) 431-1128

CIRCLE #132 ON READER SERVICE CARD.

Two new programs from Artworx

Artworx, the software publisher best known for its infamous *Strip Poker*, has released two new games, one of which follows in *Strip Poker*'s footsteps. *Centerfold Squares* is an adult game where you and your computer opponent (a centerfold girl) must battle for strategically placed squares, each of which cover a portion of the young lady's anatomy. The graphics were produced using digitization, along with some touch-up work by an artist. *Centerfold Squares* is priced at $29.95.



Also newly available from Artworx is a program that is somewhat more conventional, *Daily Double Horse Racing*. This racetrack simulation boasts complete racing forms that include histories of 180 horses and 12 jockeys, competing in almost 400 races. It's up to you to handicap the races using the information you've acquired on the horses, jockeys and track conditions. Full-featured betting is also included. *Daily Double Racing* lists for $29.95.

Artworx
1844 Penfield Road
Penfield, NY 14526
(716) 385-6120

CIRCLE #134 ON READER SERVICE CARD.

Upcoming Atari Show!

The Michigan Atari Computer Enthusiasts (MACE) has announced The Michigan Atari Computer Expo to be held on May 6 and 7, 1989 at the Detroit Metro Airport Hilton, located in Romulus, Michigan. MACE, one of the nation's oldest and largest Atari Users' Groups, was the first club to sponsor an Atari-only show with the AtariFest of 1984.

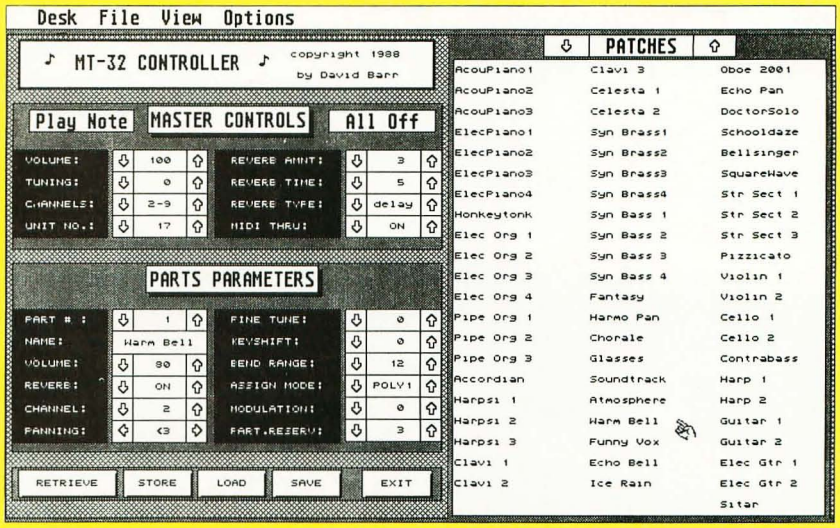
"We plan on filling over 40 booths with developers, retailers and dealers—both large and small," said Pattie Rayl, MACE coordinator. "This show is planned with the Atari user (and users' groups) in mind. Whether you have an 8-bit or an ST, you'll find lots to interest you."

The last show in the Detroit area, the MAGIC Show, was praised by exhibitors and attendees alike, and MACE plans to continue the tradition.

For information on MACE, the Michigan Atari Computer Expo, booth prices, admission rates and discount airfare, call Pattie Rayl at (313) 973-8825 or write for an exhibitor or users' group package to 3487 Braeburn Circle, Ann Arbor, MI 48108.

CIRCLE #133 ON READER SERVICE CARD.

MIDI Software



Owners of the Roland MT-32 synthesizer may be able to make use of PolySoft's newly released *MT-32 Controller*, a desk accessory that may be called up without exiting the currently running application. All the program's parameters, including MIDI channel, tuning, patch choice and panning, may be set using the mouse, and all patches and configurations may be saved to or loaded from disk.

The *MT-32 Controller* is available now and costs $49.95 plus $4.50 for postage and handling.

PolySoft
P.O. Box 69541
Portland, OR 97201
(503) 245-3152

CIRCLE #135 ON READER SERVICE CARD.

Make the DELPHI connection

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

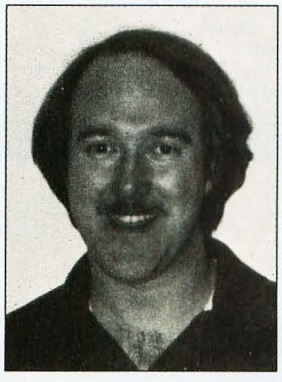
To Join DELPHI

1. Dial 617-576-0862 with any terminal or PC and modem (at 2400 bps, dial 576-2981).
2. At the Username prompt, type JOINDELPHI.
3. At the Password prompt, enter STLOG.

For more information, call DELPHI Member Services at 1-800-544-4005, or at 617-491-3393 from within Massachusetts or from outside the U.S.

DELPHI is a service of Cenberal Videotex Corporation of Cambridge, Massachusetts.

Database DELPHI



BY ANDY EDDY

Now that we are on the homestretch to summer, many people have a full slate of chores and recreational events: the final touches of spring cleaning, warming up for the local softball league, waxing the car, etc. This only means that there is that much less time for telecomputing. For this reason, we have to become more efficient in our sessions, so we still have time to do all the other things that hold our attention this time of year.

Last month we started talking about the Using DELPHI section in an effort to help you make the most of your use of the service; this month we'll finish up our coverage of this helpful area. This menu is located off the main menu, and Figure 1 shows what it will look like.

If you aren't at the main menu, you can get to the Using DELPHI area by typing "GO USI."

Let your fingers do the walking

The Past Bills/Invoices selection lets you review your bills for the last 12 months. This is particularly helpful if you have a number of associate accounts and want to keep track of the overall account usage, such as a business that uses DELPHI to keep in touch with its salespeople. This information is cursory, giving daytime and evening-time usage recaps, as well as surcharged premium-service usage. Similar to the Past Bills/Invoices selection, the Review Bills/Invoices choice lets you look through the latest bill for DELPHI usage.

Speaking of premium services, the next selection on the Using DELPHI menu, aptly titled Premium Services, details the costs for the extra-charge segments of DELPHI. The menu is brief, but contains a lot of information as you can see in Figure 2.

As we noted in *Database DELPHI* in the March 1989 issue of ST-LOG, Dialog is a service available on DELPHI that lets you search online through various magazines, newspapers and other research materials for even the slightest tidbit of data. It uses a different user interface than the rest of the DELPHI system, so they've included a sample session in the Premium Services menu that will show you how to use Di-

alog to your benefit—but without breaking the bank, because searches can cost up to $300 an hour! The Files and Rates on the Dialog selection is handy for searching through the various databases available through Dialog and for finding their per-hour costs.

The Premium Services menu also contains rate data on FAXing or telexing messages through DELPHI, as well as surcharge info for using the various financial segments of the system. These include access to stock quotes, classifieds and the News-A-Tron Market Reports, among others. The VESTOR financial service has its own information packet under the Premium Services menu.

Rates and Prices is fairly self-explanatory: It gives the up-to-date charges for using DELPHI, depending on what network you use to access it, when you call, and your discount if you are an Advantage Plan member (which we discussed in the March 1989 column). This menu selection also goes over many of the extra-cost items on DELPHI.

A choice off the Rates and Prices menu called Policy Notes is also a "must" to read. It will update you on the basics of the system, such as what times are considered office time and home time (for rate changes), what holidays DELPHI discounts usage on, various billing data and workspace charges.

We discussed the Settings menu in the July 1988 column. This menu can actually be accessed from any SIG, such as the ST-LOG SIG, by typing "SET SET." This menu lets you check on your operating status, such as where you default to after log-in, screen length and width, even your prompt character. This is also where you can change your password, which is something that you should do every once in a while for safety's sake.

Another helpful area is the Software Changes menu, which is shown in Figure 3.

DELPHI is constantly growing and is constantly changing the programming of the system to help its growth. You can scan through some of the most recently documented changes to see how it might affect your use of DELPHI. For example, if you want to see all the changes made to the Forum, you would type "PRO" (short for "PROGRAM COMPONENT"), then type "forum" as the keyword. You could then READ each listing, or, to save time, SCAN for items of particular interest.

Telex-Codes lets you pick the desired country and get the code number to facilitate your sending a message properly. Keywords can be entire country names (or pieces thereof) for wide searches as seen in Figure 4.

Terms of Agreement is simply a copy of the "contract" that you agree to as a member of DELPHI. It assures that you understand all the requirements of being an account holder, such as payment, credit, age limits, etc.

Update Credit Card is another easy-to-understand process, and is important that you keep DELPHI informed of any changes or updates to your credit card, so it can assure that there are no lapses in your DELPHI account. The menu prompts lead you through what type of card you are using, the expiration date and other related information.

Usage History provides a detailed listing of your account, rather than the simplified data on the Past Bills/Invoices selection. You'll be prompted for a user ID (you can check only your own and your associate IDs' accounts) and a date to display. What you'll get back is second-by-second list of where you went while you were online. If you suspect that someone is using your accout, this is a good way to verify what sections were accessed.

FIGURE 1

```
USING-DELPHI Menu:

Advantage Plan              Premium Services
Advice From DELPHI          Rates and Prices
Boston Membership Plan      Review Bills/Invoices
Change Address              Settings (PROFILE)
Credit Policy               Software Changes
Feedback                    Telex-Codes
Guided Tour                 Terms of Agreement
Index                       Update Credit Card
Mail To SERVICE             Usage History
Manuals                     What's New On DELPHI
Member Service              Worldwide Access Info
Network Info                HELP
Past Bills/Invoices         EXIT
```

FIGURE 2

```
PREMIUM SERVICES Menu:

Dialog Example                          Telex Rates
FAX Rates                               VESTOR surcharges
Files and Rates on Dialog               Help
Financial Information Surcharges        Exit
```

FIGURE 3

```
To see all changes, enter DATE BEFORE TOMORROW.  To seach for a specific
change, select the TYPE of change (NEW, BUG, ALL) or the PROGRAM (FORUM,
CONFERENCE, DATABASE, etc.) or a MISCELLANEOUS keyword (such as DOWNLOAD or
YMODEM).


Search Criteria Menu:

TYPE OF CHANGE
PROGRAM COMPONENT
MISCELLANEOUS
DATE OF CHANGE

Search on what criteria?
```

FIGURE 4

```
Enter country to search for, * for entire list.

Search for> mon

MONACO                  842
MONTSERRAT              360
SOLOMON ISLANDS         769
```

As we mentioned previously, DELPHI is growing, and there's no better way to keep up with the changes than the What's New On DELPHI section. You can READ through all the items on the What's New listing, or, better yet, SCAN through the article names and pick items of interest.

The last choice on the Using DELPHI menu is Worldwide Access Info. If you decide to link up to DELPHI from somewhere outside the United States, this selection will clue you in to what you might expect for surcharges and who to contact for more information.

Any questions? As always, if you don't understand how a certain feature works, you can drop a note to me in Electronic Mail at user ID ANALOG2. For a source of DELPHI help on your bookshelf, you should pick up a copy of Michael A. Banks' *DELPHI: The Official Guide*. You can find it at most major bookstores, or you can order it online by typing "GO GUIDE." Of course, the accompanying sidebar will tell you how to sign up for DELPHI *and* get you the book for a low combined cost.

Till next month, C U online.

The reviews are in . . .

"'A Best Buy' I'm impressed"
David H. Ahl, Atari Explorer, Nov-Dec 1987

"If you've got an Atari, you probably need this program."
Jerry Pournell, Byte Magazine, October 1987

"pc-ditto is a winner."
Charlie Young, ST World, July 1987

"This is the product we have been looking for."
Donna Wesolowski, ST Informer, August 1987

"This truly incredible software emulator really works."
Mike Gibbons, Current Notes, September 1987

NOW! RUN THESE IBM PROGRAMS ON YOUR ATARI ST.

Lotus 1-2-3	Flight Simulator	Framework	Symphony
Enable	Ability	DESQview	Q&A
Sidekick	Superkey	Norton Utilities	dBase II, III, III+
Crosstalk IV	Carbon Copy	Chart-Master	Print Shop
EasyCAD	DAC Easy Accounting	BPI Accounting	Turbo Pascal
GW Basic	Managing Your Money	Silvia Porter's	pfs:Professional File

And Hundreds More!

---

pc-ditto is a software-only utility which taps the power of our Atari St to imitate an IBM PC XT. No extra hardware is required (an optional 5.25-inch drive may be required for 5.25-inch disks). All your IBM disks will work "out-of-the-box".

pc-ditto features include:

- All ST models supported (520, 1040, & Mega)
- up to 703K usable memory (1040 & Mega)
- not copy-protected -- installable on hard disk
- imitates IBM monochrome and IBM color graphics adapters
- access to hard disk, if hard disk used
- optionally boots DOS from hard disk
- parallel and serial ports fully supported
- supports 3.5-inch 720K format and 360K single-sided formats
- supports optional 5.25-inch 40-track drives

System requirements:

- IBM PC-DOS or Compaq MS-DOS version 3.2 or above recommended
- optional 5.25-inch drive is required to use 5.25-inch disks
- 3.5-inch 720K DOS disks require a double-sided drive (Atari SF314 or equivalent)

See pc-ditto today at an Atari dealer near you, or write for free information!

pc-ditto
by
Avant-Garde Systems
381 Pablo Point Drive
Jacksonville, FL 32225
(904) 221-2904

$89.95

Avant-Garde Systems, 381 Pablo Point Dr.
Jacksonville, Florida 32225 (904) 221-2904
Yes! Please send information on pc-ditto.
Name_____
Address_____
City_____ State_____ Zip_____

CIRCLE #104 ON READER SERVICE CARD.

Line Attack

by Kirk Stover

Since the 520ST hit the market, there have been very few action games available where two people can play against each other at the same time. Here's a game of speed and skill for two players that's a lot of fun to play.

In *Line Attack!* the object is to draw the longest line possible, using right-angle turns. The longer your line survives, the more points you accumulate. You will crash whenever you run into your opponent's line, run into the border or run into your own line. At the same time you're trying to add points to your score, you try to make your opponent crash by forcing him to turn. You may move up, down, left and right. If you try to reverse on your own line, you will crash. If both players run into each other head-on, both will crash, which results in a tie. The round ends with the first crash, and the winner receives all of his points, while the loser receives no points at all. The action can be paused at any time by pressing the fire button and resumed by pressing the fire button again.

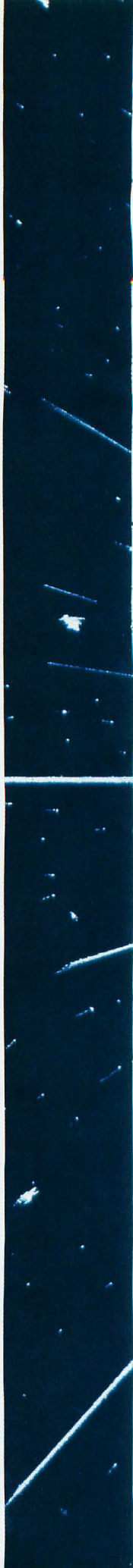
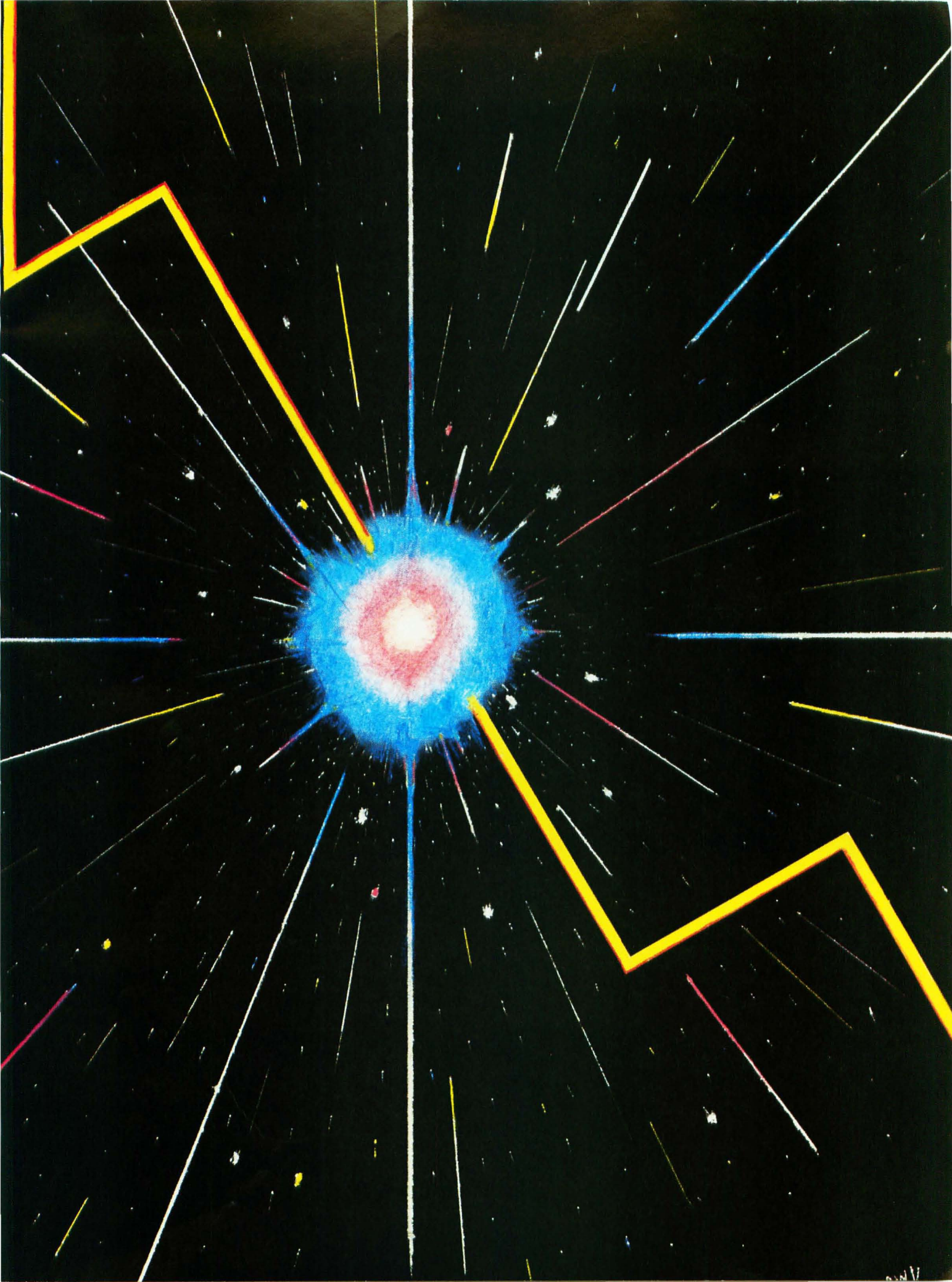


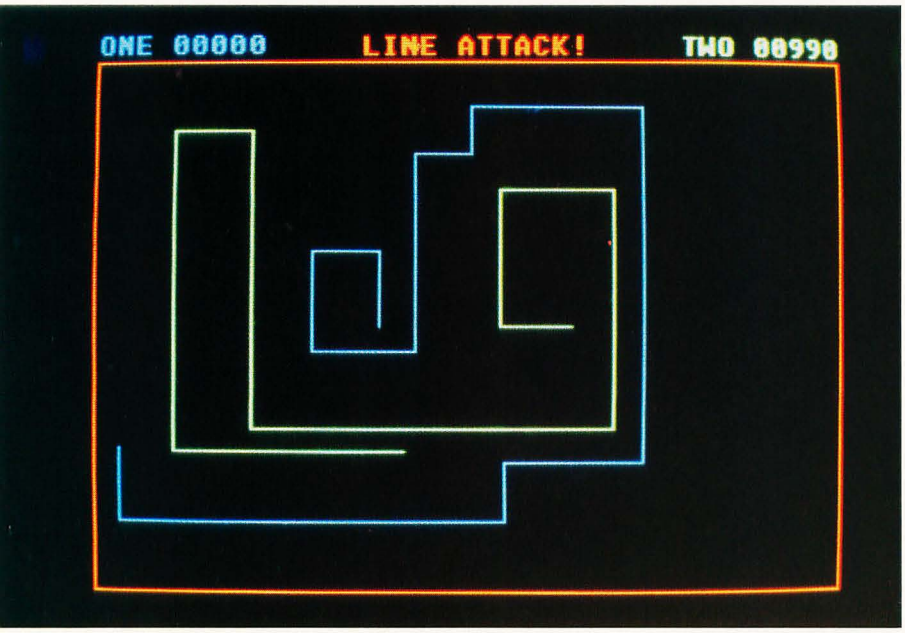
ILLUSTRATION • AWD



To play *Line Attack!*, you'll need two joysticks and a color monitor in low resolution mode. To begin, simply double-click on LNATTACK.PRG. You will then be presented with an option screen to choose which skill level you want to play and how many rounds per game. Use Joystick 1 to control the options. The bar indicator represents the relative skill. The further to the right you move the bar, the more difficult the game. This is achieved by moving the joystick up or down. The rounds per game are determined by moving the joystick left or right. When the bar indicator is at the far left you will have one round of play. If you move the bar to the far right, you will have nine rounds of play. After your selections have been made, you start the game by pressing the fire button. However, if you wish to quit, you can press the Undo key.

In Line Attack! the object is to draw the longest line possible, using right-angle turns. The longer your line survives, the more points you accumulate.

After pressing the fire button the playing screen will be displayed with two dots on it. Joystick 1 is the blue player, and Joystick 2 is the green. Your lines grow from these two dots. There is a status line at the top of the screen that displays both players' scores. At the start of each round, the status line displays the current round number with a prompt to begin. Press the fire button to begin, but be ready—*Line Attack!* is a fast-moving game.



A feature that makes it easier to play is that you don't have to hold the joystick in the direction you want to go in order to keep moving. Once you turn, you can release the joystick, and the line will continue in that direction until you turn again. Be careful when moving your joystick, as diagonal moves are ignored.

When the round is finished, the status line will flash the winning player. There will then be a short delay, and the new screen for the next round will appear. After the final round is finished, the winner of the game is declared, and you press the fire button to return to the option screen in order to play again or quit.

Line Attack! was written in assembly language and was challenging to write because I found myself working with several things I had only used separately in the past. For instance, Line-A routines were used to place the pixels and determine collisions. Colors, text and graphics were accomplished using the VDI. The AES was used for growing and shrinking boxes to simulate explosions. The intelligent keyboard was placed into the joystick interrogate mode in order to read the joysticks.

The real fun, though, is playing it. I'll warn you: *Line Attack!* is definitely habit-forming! ▪

Kirk Stover has had many programs published in ST-LOG, *including* Dr. Flop E. Disk *in Issue 10,* CZ-101 Patch Librarian *from Issue 14 and* Super Spool *from Issue 28. He lives in Minnesota.*

LINE ATTACK
Listing 1:
Assembly

Line Attack by Kirk Stover

```
*************************************************************************
******                                                             ******
******                        Line Attack!                         ******
******                            by                               ******
******                        Kirk Stover                          ******
******                   Copyright 1989 by ST-LOG                  ******
******                                                             ******
*************************************************************************

                    text

line_A_initialise   equ   $a000
line_A_putpixel     equ   $a001
line_A_getpixel     equ   $a002
line_A_line         equ   $a003

gemdos              equ   1
term                equ   0
rawconio            equ   6
conws               equ   9
setblock            equ   74

gem                 equ   2

bios                equ   13
bconout             equ   3

xbios               equ   14
initmous            equ   0
dosound             equ   32
kbdvbase            equ   34

stick               equ   0
xloc                equ   2
yloc                equ   4
score               equ   6
kolor               equ   8

*---------------------

main            move.l    sp,a5                 ; save stack pointer
                move.l    #mystack,sp           ; and point to our stack
                move.l    4(a5),a5              ; base page start pointer
                move.l    #$100,d0              ; length of base page
                add.l     $0c(a5),d0            ; length of text section
                add.l     $14(a5),d0            ; length of data section
                add.l     $1c(a5),d0            ; length of bss section
                move.l    d0,-(sp)              ; amount of memory to reserve
                move.l    a5,-(sp)              ; starting address of memory
                move.w    #0,-(sp)              ; dummy word
                move.w    #setblock,-(sp)       ; reserve memory
                trap      #gemdos               ; call gemdos
                add.l     #12,sp                ; restore stack
                bsr       open_gem              ; open virtual workstation
                bsr       initialise            ; do housework
                bsr       new_game              ; let's go!
                bsr       terminate             ; more housework
exit            bsr       close_gem             ; close virtual workstation
                move.w    #term,-(sp)           ; back to desktop
                trap      #gemdos               ; call gemdos

*---------------------

call_aes        move.l    #aespb,d1             ; pointer to aes tables
                move.w    #$c8,d0               ; function code
                trap      #gem                  ; call gem
                rts                             ; return

call_vdi        move.l    #vdipb,d1             ; pointer to vdi tables
                move.w    #$73,d0               ; function code
                trap      #gem                  ; call gem
                rts                             ; return

*---------------------

open_gem        moveq.l   #0,d0                 ; use D0 as zero
                move.l    d0,ap1resv            ; zero out reserved parameters
                move.l    d0,ap2resv
                move.l    d0,ap3resv
                move.l    d0,ap4resv
                move.w    #10,opcode            ; appl_init function
                move.w    d0,sintin
                move.w    #1,sintout
                move.w    d0,saddrin
                move.w    d0,saddrout
```

PROGRAM LISTINGS

Line Attack

by Kirk Stoever

```
                    bsr     call_aes

                    move.w  intout,appl_id        ; save appl_id
                    move.w  #77,opcode            ; graf_handle function
                    move.w  #0,sintin
                    move.w  #5,sintout
                    move.w  #0,saddrin
                    move.w  #0,saddrout
                    bsr     call_aes
                    move.w  intout,graf_handle    ; save graf_handle

                    move.w  #100,contrl           ; v_opnvwk function
                    move.w  #0,contrl+2
                    move.w  #11,contrl+6
                    move.w  graf_handle,contrl+12
                    move.w  #1,intin
                    move.w  #1,intin+2
                    move.w  #1,intin+4
                    move.w  #1,intin+6
                    move.w  #1,intin+8
                    move.w  #1,intin+10
                    move.w  #1,intin+12
                    move.w  #1,intin+14
                    move.w  #1,intin+16
                    move.w  #1,intin+18
                    move.w  #2,intin+20
                    bsr     call_vdi
                    move.w  contrl+12,vdi_handle ; save vdi_handle

                    cmp.w   #16,intout+26         ; are we in low-resolution?
                    beq     open_gemx             ;   yes, go exit
                    move.w  #52,opcode            ;   no, display form_alert
                    move.w  #1,sintin
                    move.w  #1,sintout
                    move.w  #1,saddrin
                    move.w  #0,saddrout
                    move.w  #1,intin
                    move.l  #res_alert,addrin
                    bsr     call_aes
                    move.l  #exit,(sp)            ; substitute exit address
open_gemx           rts

*---------------------

close_gem           move.w  #101,opcode           ; v_clsvwk
                    move.w  #0,contrl+2
                    move.w  #0,contrl+4
                    move.w  #0,contrl+6
                    move.w  #0,contrl+8
                    move.w  vdi_handle,contrl+12
                    bsr     call_vdi
                    rts

*---------------------

v_clrwk             move.w  #3,contrl
                    move.w  #0,contrl+2
                    move.w  #0,contrl+6
                    move.w  vdi_handle,contrl+12
                    bsr     call_vdi
                    rts

*---------------------

vst_color           move.w  #22,contrl
                    move.w  #0,contrl+2
                    move.w  #1,contrl+6
                    move.w  vdi_handle,contrl+12
                    move.w  d0,intin
                    bsr     call_vdi
                    rts

*---------------------

vst_effects         move.w  #106,contrl
                    move.w  #0,contrl+2
                    move.w  #1,contrl+6
                    move.w  vdi_handle,contrl+12
                    move.w  d0,intin
                    bsr     call_vdi
                    rts

*---------------------

vst_height          move.w  #12,contrl
```

Line Attack by Kirk Stoever

```
                    move.w   #1,contrl+2
                    move.w   #0,contrl+6
                    move.w   vdi_handle,contrl+12
                    move.w   #0,ptsin
                    move.w   d0,ptsin+2
                    bsr      call_vdi
                    rts

*--------------------

vg_text             move.w   #8,contrl
                    move.w   #1,contrl+2
                    move.w   vdi_handle,contrl+12
                    move.w   d0,ptsin
                    move.w   d1,ptsin+2
                    clr.w    d1
                    move.l   #intin,a1
                    clr.w    d0
vg_text_1           move.b   (a0)+,d0
                    beq      vg_text_2
                    move.w   d0,(a1)+
                    addq.w   #1,d1
                    bra      vg_text_1
vg_text_2           move.w   d1,contrl+6
                    bsr      call_vdi
                    rts

*--------------------

vsf_color           move.w   #25,contrl
                    move.w   #0,contrl+2
                    move.w   #1,contrl+6
                    move.w   vdi_handle,contrl+12
                    move.w   d0,intin
                    bsr      call_vdi
                    rts

*--------------------

vsf_interior        move.w   #23,contrl
                    move.w   #0,contrl+2
                    move.w   #1,contrl+6
                    move.w   vdi_handle,contrl+12
                    move.w   d0,intin
                    bsr      call_vdi
                    rts

*--------------------

vs_color            move.w   #14,contrl
                    move.w   #0,contrl+2
                    move.w   #4,contrl+6
                    move.w   vdi_handle,contrl+12
                    move.w   #1,intin
                    move.w   (a0)+,intin+2        ; a0 points to rgb value
                    move.w   (a0)+,intin+4
                    move.w   (a0),intin+6
                    bsr      call_vdi
                    rts

*--------------------

vswr_mode           move.w   #32,contrl
                    move.w   #0,contrl+2
                    move.w   #1,contrl+6
                    move.w   vdi_handle,contrl+12
                    move.w   d0,intin
                    bsr      call_vdi
                    rts

*--------------------

v_bar               move.w   #11,contrl
                    move.w   #2,contrl+2
                    move.w   #0,contrl+6
                    move.w   #1,contrl+10
                    move.w   vdi_handle,contrl+12
                    bsr      call_vdi
                    rts

*--------------------

mouse_off           move.w   #78,opcode
                    move.w   #1,sintin
                    move.w   #1,sintout
```

PROGRAM LISTINGS

Line Attack

by Kirk Stoever

```
                move.w   #1,saddrin
                move.w   #0,saddrout
                move.w   #256,intin
                bsr      call_aes
                rts

*-------------------

save_color      move.w   #0,d3
                move.l   #old_color,a3
save_color_1    move.w   #26,contrl          ; vq_color function
                move.w   #0,contrl+2
                move.w   #2,contrl+6
                move.w   vdi_handle,contrl+12
                move.w   d3,intin
                move.w   #0,intin+2
                bsr      call_vdi
                move.w   intout+2,(a3)+
                move.w   intout+4,(a3)+
                move.w   intout+6,(a3)+
                add.w    #1,d3
                cmp.w    #5,d3
                bne      save_color_1
                rts

*-------------------

set_color       move.w   #0,d3
set_color_1     move.w   #14,contrl          ; vs_color function
                move.w   #0,contrl+2
                move.w   #4,contrl+6
                move.w   vdi_handle,contrl+12
                move.w   d3,intin
                move.w   (a3)+,intin+2
                move.w   (a3)+,intin+4
                move.w   (a3)+,intin+6
                bsr      call_vdi
                add.w    #1,d3
                cmp.w    #5,d3
                bne      set_color_1
                rts

*-------------------

initialise      bsr      mouse_off       ; turn the critter off
                bsr      save_color      ; save current color values
                move.l   #new_color,a3   ; point to our colors
                bsr      set_color       ; and use them

                move.w   #kbdvbase,-(sp) ; kbdvbase function
                trap     #xbios          ; call xbios
                addq.l   #2,sp           ; restore stack
                move.l   d0,a0           ; set a0 to returned address
                move.l   a0,save_kbdvbase ; save vector table address
                move.l   24(a0),old_joy  ; save old joystick vector
                move.l   #joystick,24(a0) ; substitute our routine

                move.w   #$15,-(sp)      ; interogate mode
                move.w   #4,-(sp)        ; ikbd
                move.w   #bconout,-(sp)  ; bconout function
                trap     #bios           ; call bios
                addq.l   #6,sp           ; restore stack

                move.w   #0,flagged      ; turn off joystick flag
                rts

*-------------------

terminate       move.l   #old_color,a3   ; point to saved colors
                bsr      set_color       ; and use them
                move.w   #$1a,-(sp)      ; joystick scan off mode
                move.w   #4,-(sp)        ; ikbd
                move.w   #bconout,-(sp)  ; bconout function
                trap     #bios           ; call bios
                addq.l   #6,sp           ; restore stack
                move.l   save_kbdvbase,a0 ; point to vector table
                move.l   old_joy,24(a0)  ; restore old joystick vector
                move.l   16(a0),-(sp)    ; mouse vector
                move.l   #mousedata,-(sp) ; mouse parameters
                move.w   #1,-(sp)        ; enable mouse, relative mode
                move.w   #initmous,-(sp) ; initmous function
                trap     #xbios          ; call xbios
                add.l    #12,sp          ; restore stack
                rts
```

Line Attack

by Kirk Stoever

```
*---------------------

joystick        tst.w   flagged         ; have we read the joystick yet?
                bne     joystick_x      ;   no, go exit
                move.b  0(a0),joy_rec   ;   yes, save new values
                move.b  1(a0),joy_rec+1
                move.b  2(a0),joy_rec+2
                move.w  #1,flagged      ; turn on indicator
joystick_x      rts

*---------------------

read_joy        move.w  #$16,-(sp)      ; interrogate joystick
                move.w  #4,-(sp)        ; ikbd
                move.w  #bconout,-(sp)  ; bconout function
                trap    #bios           ; call bios
                addq.l  #6,sp           ; restore stack
read_joy_1      tst.w   flagged         ; is joystick packet available?
                beq     read_joy_1      ;   no, wait for one
                move.w  #0,flagged      ; reset flag
                lea     joy_rec,a0      ; point to joystick record
                move.b  (a0),d0         ; joystick one value
                and.w   #$80,d0         ; isolate fire button status
                move.w  d0,fire_button  ; and save it
                move.b  (a0)+,d0        ; joystick one value
                and.w   #$0f,d0         ; isolate direction status
                cmp.b   #1,d0           ; valid direction?
                beq     read_joy_2      ;   yes
                cmp.b   #2,d0           ; valid direction?
                beq     read_joy_2      ;   yes
                cmp.b   #4,d0           ; valid direction?
                beq     read_joy_2      ;   yes
                cmp.b   #8,d0           ; valid direction?
                bne     read_joy_3      ;   no, leave current status unchanged
read_joy_2      move.w  d0,player1      ; save new player 1 direction
read_joy_3      move.b  (a0),d0         ; joystick two value
                and.w   #$0f,d0         ; isolate direction status
                cmp.b   #1,d0           ; valid direction?
                beq     read_joy_4      ;   yes
                cmp.b   #2,d0           ; valid direction?
                beq     read_joy_4      ;   yes
                cmp.b   #4,d0           ; valid direction?
                beq     read_joy_4      ;   yes
                cmp.b   #8,d0           ; valid direction?
                bne     read_joy_5      ;   no, leave current status unchanged
read_joy_4      move.w  d0,player2      ; save new player 2 direction
read_joy_5      move.b  (a0),d0         ; joystick two value
                and.w   #$80,d0         ; isolate fire button status
                or.b    d0,fire_button  ; and logical OR with joystick 1
                rts

*---------------------

no_fire         bsr     read_joy        ; read joystick
                tst.w   fire_button     ; is fire button pressed?
                bne     no_fire         ;   yes, go try again
                rts

*---------------------

delay           move.w  #0,d1           ; enter with D0 set for number
delay_1         sub.w   #1,d1           ; of iterations
                bne     delay_1
                sub.w   #1,d0
                bpl     delay_1
                rts

*---------------------

show_score      move.l  #player1,a5     ; point to player 1
                move.w  score(a5),d0    ; move the score
                move.l  #score_one,a0   ; point to ascii conversion area
                bsr     bin_to_dec      ; and convert
                move.l  #player2,a5     ; point to player 2
                move.w  score(a5),d0    ; move the score
                move.l  #score_two,a0   ; point to ascii conversion area
                bsr     bin_to_dec      ; and convert
                move.l  #score_msg,a0   ; point to string
                bsr     write_str       ; and print it
                rts

*---------------------

bin_to_dec      addq.l  #5,a0           ; point to end of conversion area
                move.w  #4,d1           ; loop counter 5 digits - 1
```

```
bin_to_dec_1    ext.l    d0               ; extend the sign
                divs     #10,d0           ; divide by decimal ten
                swap     d0               ; use remainder
                move.b   d0,-(a0)         ; and store in string
                add.b    #'0',(a0)        ; convert to ascii
                swap     d0               ; fix register
                dbra     d1,bin_to_dec_1  ; and loop
                rts

*---------------------

pause           bsr      no_fire          ; wait for fire button to be released
pause_1         bsr      read_joy         ; read joystick
                tst.w    fire_button      ; is fire button pressed again?
                beq      pause_1          ;   no, try again
                bsr      no_fire          ;   yes, wait for release
                rts

*---------------------

flash_msg       move.l   a3,a0            ; move string1 pointer to a0
                bsr      write_str        ; and print it
                bsr      read_joy         ; read joystick
                tst.w    fire_button      ; is fire button pressed?
                bne      flash_msg_x      ;   yes, go exit
                move.w   #1,d0            ; delay for
                bsr      delay            ; a while
                bsr      read_joy         ; read joystick
                tst.w    fire_button      ; is fire button pressed?
                bne      flash_msg_x      ;   yes, go exit
                move.l   a4,a0            ; move string2 pointer to a0
                bsr      write_str        ; and print it
                move.w   #1,d0            ; delay for
                bsr      delay            ; a while
                bra      flash_msg        ; go loop
flash_msg_x     rts

*---------------------

draw_screen     bsr      v_clrwk
                move.w   #73,opcode       ; graf_growbox
                move.w   #8,sintin
                move.w   #1,sintout
                move.w   #0,saddrin
                move.w   #0,saddrout
                move.w   #155,intin
                move.w   #96,intin+2
                move.w   #8,intin+4
                move.w   #6,intin+6
                move.w   #0,intin+8
                move.w   #0,intin+10
                move.w   #320,intin+12
                move.w   #200,intin+14
                bsr      call_aes
                lea      title_msg,a0     ; point to title string
                bsr      write_str        ; and print it
                bsr      show_score       ; print the current score

                dc.w     line_A_initialise
                move.l   a0,line_A_var    ; store line A variable pointer
                move.l   8(a0),intin_addr
                move.l   12(a0),ptsin_addr
                move.l   line_A_var,a0    ; restore line A pointer
                move.w   #000,38(a0)      ; x1 coordinate
                move.w   #009,40(a0)      ; y1 coordinate
                move.w   #319,42(a0)      ; x2 coordinate
                move.w   #009,44(a0)      ; y2 coordinate
                bsr      draw_line
                move.l   line_A_var,a0    ; restore line A pointer
                move.w   #319,38(a0)      ; x1 coordinate
                move.w   #009,40(a0)      ; y1 coordinate
                move.w   #319,42(a0)      ; x2 coordinate
                move.w   #199,44(a0)      ; y2 coordinate
                bsr      draw_line
                move.l   line_A_var,a0    ; restore line A pointer
                move.w   #000,38(a0)      ; x1 coordinate
                move.w   #199,40(a0)      ; y1 coordinate
                move.w   #319,42(a0)      ; x2 coordinate
                move.w   #199,44(a0)      ; y2 coordinate
                bsr      draw_line
                move.l   line_A_var,a0    ; restore line A pointer
                move.w   #000,38(a0)      ; x1 coordinate
                move.w   #009,40(a0)      ; y1 coordinate
                move.w   #000,42(a0)      ; x2 coordinate
                move.w   #199,44(a0)      ; y2 coordinate
```

Line Attack

by kirk stoever

```
                bsr     draw_line
                rts

*---------------------

draw_line       move.w  #0,24(a0)       ; draw red border
                move.w  #0,26(a0)
                move.w  #1,28(a0)
                move.w  #0,30(a0)
                move.w  #$ffff,32(a0)   ; last line
                move.w  #$ffff,34(a0)   ; line mask - solid
                move.w  #000,36(a0)     ; write mode - replace
                dc.w    line_A_line
                rts

*---------------------

draw_pixel      move.l  a5,-(sp)        ; save player pointer
                move.l  intin_addr,a3
                move.l  ptsin_addr,a4
                move.w  xloc(a5),(a4)   ; point to x position
                move.w  yloc(a5),2(a4)  ; point to y position
                move.w  kolor(a5),(a3)  ; color of pixel
                dc.w    line_A_putpixel
                move.l  (sp)+,a5        ; restore player pointer
                add.w   #1,points       ; add to # of pixels drawn
                rts

*---------------------

get_pixel       move.l  a5,-(sp)        ; save player pointer
                move.l  ptsin_addr,a4
                move.w  xloc(a5),(a4)   ; point to x position
                move.w  yloc(a5),2(a4)  ; point to y position
                dc.w    line_A_getpixel
                move.l  (sp)+,a5        ; restore player pointer
                rts                     ; return color value in D0

*---------------------

write_str       move.l  a0,-(sp)        ; string pointer
                move.w  #conws,-(sp)    ; write string function
                trap    #gemdos         ; call gemdos
                addq.l  #6,sp           ; restore stack
                rts

*---------------------

shrink_box      move.l  #shrink_noise,-(sp)
                move.w  #dosound,-(sp)  ; output sound string
                trap    #xbios          ; call xbios
                addq.l  #6,sp           ; restore stack
                move.w  #74,opcode      ; graf_shrinkbox
                move.w  #8,sintin
                move.w  #1,sintout
                move.w  #0,saddrin
                move.w  #0,saddrout
                bsr     call_aes
                move.w  #0,d0           ; short delay
                bsr     delay
                rts

*---------------------

grow_box        move.l  #explode_noise,-(sp)
                move.w  #dosound,-(sp)  ; output sound string
                trap    #xbios          ; call xbios
                addq.l  #6,sp           ; restore stack
                move.w  #73,opcode      ; graf_growbox
                move.w  #8,sintin
                move.w  #1,sintout
                move.w  #0,saddrin
                move.w  #0,saddrout
                bsr     call_aes
                move.l  #new_color+6,a0 ; point to color 1
                bsr     vs_color        ; and restore
                rts

*---------------------

box_color       move.w  kolor(a5),d0    ; player color value
                add.w   #1,d0           ;
                mulu    #6,d0           ; calculate color offset
                move.l  #new_color,a0   ; and add to
                add.w   d0,a0           ; color table pointer
```

Line Attack

by Kirk Stoever

```
                bsr     vs_color        ; go set color
                rts

*--------------------

explode         bsr     box_color       ; set player explode color
                move.w  xloc(a5),intin  ; starting x coordinate
                move.w  yloc(a5),intin+2 ; starting y coordinate
                move.w  #1,intin+4      ; starting width
                move.w  #1,intin+6      ; starting height
                move.w  xloc(a5),d0     ; current x position
                cmp.w   #29,d0          ; are we at left border?
                bgt     explode_1       ;   no
                move.w  #0,d0           ; use left border
                bra     explode_2       ; as new end x coord
explode_1       sub.w   #30,d0          ; calculate new x coord
explode_2       move.w  d0,intin+8      ; ending x coordinate
                move.w  yloc(a5),d1     ; current y position
                cmp.w   #29,d1          ; are we at top border?
                bgt     explode_3       ;   no
                move.w  #9,d1           ; use top border
                bra     explode_4       ; as new end y coord
explode_3       sub.w   #20,d1          ; calculate new y coord
explode_4       move.w  d1,intin+10     ; ending y coordinate
                cmp.w   #260,d0         ; are we at right border?
                blt     explode_5       ;   no
                sub.w   #319,d0         ; calculate ending
                neg.w   d0              ; width
                move.w  d0,intin+12     ; and save
                bra     explode_6
explode_5       move.w  #60,intin+12    ; save ending width
explode_6       cmp.w   #160,d1         ; are we at bottom border?
                blt     explode_7       ;   no
                sub.w   #200,d1         ; calculate ending
                neg.w   d1              ; height
                move.w  d1,intin+14     ; and save
                bra     explode_8
explode_7       move.w  #40,intin+14    ; save ending height
explode_8       bsr     grow_box        ; show expanding box
                move.w  #1,d0           ; and delay for
                bsr     delay           ; awhile
                rts

*--------------------

wait_loop       move.w  skill,d0        ; use skill level as delay factor
wait_loop_1     add.w   #1,d0           ; increment counter
                cmp.w   #10,d0          ; maximum reached?
                beq     wait_loop_x     ;   yes, go exit
                move.w  #1000,d1        ; enter delay loop
wait_loop_2     dbra    d1,wait_loop_2  ;
                bra     wait_loop_1     ; try again
wait_loop_x     rts

*--------------------

find_winner     move.w  points,d5       ; divide the # of points
                asr.w   #1,d5           ; by two
                cmp.w   #1,done         ; is player 1 color = done?
                bne     find_winner_1   ;   no
                move.l  #win2_msg,a3    ; player 2 is the winner
                move.l  #player2,a5     ; point to player 2
                add.w   d5,score(a5)    ; add to the score
                move.l  #player1,a5     ; point to player 1
                bra     find_winner_x   ; go explode it
find_winner_1   cmp.w   #2,done         ; is player 2 color = done?
                bne     find_winner_2   ;   no
                move.l  #win1_msg,a3    ; player 1 is the winner
                move.l  #player1,a5     ; point to player 1
                add.w   d5,score(a5)    ; add to the score
                move.l  #player2,a5     ; point to player 2
                bra     find_winner_x   ; go explode it
find_winner_2   move.l  #tie_msg,a3     ; players tied
                move.l  #player1,a5     ; point to player 1
                add.w   d5,score(a5)    ; add to the score
                bsr     explode         ; go explode it
                move.l  #player2,a5     ; point to player 2
                add.w   d5,score(a5)    ; add to the score
find_winner_x   bsr     explode         ; explode player pointed to by a5
                bsr     show_score      ; show new score
                move.l  a3,a0           ; point to winner message
                bsr     write_str       ; and print it
                move.w  #8,d0           ; delay counter
                bsr     delay           ; wait for awhile
                rts
```

Line Attack by Kirk Stoever

```
*--------------------

skill_bar       move.w   #1,d0          ; color = white
                bsr      vsf_color      ; set fill color
                move.w   #59,ptsin      ; bar beg x coordinate
                move.w   #109,ptsin+2   ; bar beg y coordinate
                move.w   #260,ptsin+4   ; bar end x coordinate
                move.w   #121,ptsin+6   ; bar end y coordinate
                bsr      v_bar          ; draw bar
                move.w   #2,d0          ; color = blue
                bsr      vsf_color      ; set fill color
                move.w   #60,ptsin      ; bar beg x coordinate
                move.w   #110,ptsin+2   ; bar beg y coordinate
                move.w   skill,d0       ; calculate level of
                mulu     #20,d0         ; skill for bar
                add.w    #60,d0         ; indicator and use for
                move.w   d0,ptsin+4     ; bar end x coordinate
                move.w   #120,ptsin+6   ; bar end y coordinate
                bsr      v_bar          ; draw bar
                rts

*--------------------

rounds_bar      move.w   #1,d0          ; color = white
                bsr      vsf_color      ; set fill color
                move.w   #59,ptsin      ; bar beg x coordinate
                move.w   #149,ptsin+2   ; bar beg y coordinate
                move.w   #260,ptsin+4   ; bar end x coordinate
                move.w   #161,ptsin+6   ; bar end y coordinate
                bsr      v_bar          ; draw bar
                move.w   #3,d0          ; color = green
                bsr      vsf_color      ; set fill color
                move.w   #60,ptsin      ; bar beg x coordinate
                move.w   #150,ptsin+2   ; bar beg y coordinate
                move.w   rounds,d0      ; calculate number of
                mulu     #20,d0         ; rounds for bar
                add.w    #60,d0         ; indicator and use for
                move.w   d0,ptsin+4     ; bar end x coordinate
                move.w   #160,ptsin+6   ; bar end y coordinate
                bsr      v_bar          ; draw bar
                rts

*--------------------

new_game        bsr      v_clrwk        ; clear the screen
                move.w   #4,d0          ; color = red
                bsr      vst_color      ; set text color
                move.w   #5,d0          ; effects = bold and italic
                bsr      vst_effects    ; set text effects
                move.w   #60,d0         ; height = 60
                bsr      vst_height     ; set text height
                move.w   #60,d0         ; x location
                move.w   #34,d1         ; y location
                move.l   #gr_title,a0   ; title string
                bsr      vg_text        ; print graphic text
                move.w   #1,d0          ; color = white
                bsr      vst_color      ; set text color
                move.w   #0,d0          ; effects = normal
                bsr      vst_effects    ; set text effects
                move.w   #6,d0          ; height = 6
                bsr      vst_height     ; set text height
                move.w   #106,d0        ; x location
                move.w   #56,d1         ; y location
                move.l   #gr_author,a0  ; author string
                bsr      vg_text        ; print graphic text
                move.w   #2,d0          ; color = blue
                bsr      vst_color      ; set text color
                move.w   #8,d0          ; height = 8 pixels high
                bsr      vst_height     ; set text height
                move.w   #78,d0         ; x location
                move.w   #100,d1        ; y location
                move.l   #gr_skill,a0   ; skill string
                bsr      vg_text        ; print graphic text
                move.w   #47,d0         ; x location
                move.w   #119,d1        ; y location
                move.l   #gr_minus,a0   ; - string
                bsr      vg_text        ; print graphic text
                move.w   #263,d0        ; x location
                move.w   #119,d1        ; y location
                move.l   #gr_plus,a0    ; + string
                bsr      vg_text        ; print graphic text
                move.w   #3,d0          ; color = green
                bsr      vst_color      ; set text color
                move.w   #78,d0         ; x location
```

Line Attack

by Kirk Stoever

```
                move.w   #140,d1          ; y location
                move.l   #gr_rounds,a0    ; rounds string
                bsr      vg_text          ; print graphic text
                move.w   #47,d0           ; x location
                move.w   #159,d1          ; y location
                move.l   #gr_minus,a0     ; - string
                bsr      vg_text          ; print graphic text
                move.w   #263,d0          ; x location
                move.w   #159,d1          ; y location
                move.l   #gr_plus,a0      ; + string
                bsr      vg_text          ; print graphic text
                move.w   #4,d0            ; color = red
                bsr      vst_color        ; set text color
                move.w   #90,d0           ; x location
                move.w   #180,d1          ; y location
                move.l   #gr_fire,a0      ; fire string
                bsr      vg_text          ; print graphic text
                move.w   #1,d0            ; fill = solid
                bsr      vsf_interior     ; set fill interior style
                move.w   #1,d0            ; write mode = replace
                bsr      vswr_mode        ; set write mode
                bsr      skill_bar        ; draw skill bar
                bsr      rounds_bar       ; draw rounds bar
                bsr      no_fire          ; wait for no fire button

game_level      move.w   #0,player1       ; zero out joystick status
                bsr      read_joy         ; read joystick 1
                tst.w    fire_button      ; fire button pressed?
                bne      game_init        ;   yes, initialize game
                move.w   #$ff,-(sp)       ; check keyboard for characters
                move.w   #rawconio,-(sp)  ; no echo
                trap     #gemdos          ; call gemdos
                addq.l   #4,sp            ; restore stack
                swap     d0               ; check key code
                cmp.w    #$61,d0          ; is it UNDO key?
                bne      game_level_0     ;   no
                rts                       ; END OF GAME
game_level_0    move.w   #0,d0            ; wait for a
                bsr      delay            ; little while
                move.w   player1,d0       ; use joystick 1 status
                cmp.w    #1,d0            ; joystick up?
                bne      game_level_1     ;   no
                cmp.w    #9,skill         ; maximum skill?
                beq      game_level       ;   yes
                add.w    #1,skill         ; increment skill
                bsr      skill_bar        ; redraw bar
                bra      game_level       ; loop
game_level_1    cmp.w    #2,d0            ; joystick down?
                bne      game_level_2     ;   no
                cmp.w    #1,skill         ; minimum skill?
                beq      game_level       ;   yes
                sub.w    #1,skill         ; decrement skill
                bsr      skill_bar        ; redraw bar
                bra      game_level       ; loop
game_level_2    cmp.w    #4,d0            ; joystick left?
                bne      game_level_3     ;   no
                cmp.w    #1,rounds        ; minimum rounds?
                beq      game_level       ;   yes
                sub.w    #1,rounds        ; decrement rounds
                bsr      rounds_bar       ; redraw bar
                bra      game_level       ; loop
game_level_3    cmp.w    #8,d0            ; joystick right?
                bne      game_level       ;   no
                cmp.w    #9,rounds        ; maximum rounds?
                beq      game_level       ;   yes
                add.w    #1,rounds        ; increment rounds
                bsr      rounds_bar       ; redraw bar
                bra      game_level       ; loop

game_init       move.l   #player1,a5      ; point to player 1
                move.w   #0,score(a5)     ; zero out score
                move.l   #player2,a5      ; point to player 2
                move.w   #0,score(a5)     ; zero out score
                move.b   #'0',round_no    ; reset round number

new_round       bsr      draw_screen      ; create playing screen

                move.l   #player1,a5      ; point to player 1
                move.w   #8,stick(a5)     ; default stick to right
                move.w   #119,xloc(a5)    ; beginning x coord
                move.w   #104,yloc(a5)    ; beginning y coord
                bsr      box_color        ; set color for player 1
                move.w   #99,intin        ; set coordinates for shrinkbox
                move.w   #88,intin+2
                move.w   #40,intin+4
```

Line Attack by kirk stoever

```
                    move.w   #32,intin+6
                    move.w   #49,intin+8
                    move.w   #48,intin+10
                    move.w   #140,intin+12
                    move.w   #112,intin+14
                    bsr      shrink_box      ; draw shrinking box
                    bsr      draw_pixel      ; draw the pixel

                    move.w   #1,d0           ; wait for a
                    bsr      delay           ; little while

                    move.l   #player2,a5     ; point to player 2
                    move.w   #4,stick(a5)    ; default stick to left
                    move.w   #200,xloc(a5)   ; beginning x coord
                    move.w   #104,yloc(a5)   ; beginning y coord
                    bsr      box_color       ; set color for player 2
                    move.w   #180,intin      ; set coordinates for shrinkbox
                    move.w   #88,intin+2
                    move.w   #40,intin+4
                    move.w   #32,intin+6
                    move.w   #130,intin+8
                    move.w   #48,intin+10
                    move.w   #140,intin+12
                    move.w   #112,intin+14
                    bsr      shrink_box      ; draw shrinking box
                    bsr      draw_pixel      ; draw the pixel

                    add.b    #1,round_no     ; increment round number
                    move.w   #0,done         ; reset DONE flag
                    move.w   #0,points       ; reset pixel counter
                    move.w   #3,sound_count  ; reset sound counter

                    bsr      no_fire         ; wait for no fire button
                    move.l   #round_msg,a3   ; pointer to round # string
                    move.l   #fire_msg,a4    ; pointer to fire string
                    bsr      flash_msg       ; go flash messages
                    move.l   #game_msg,a0    ; point to game string
                    bsr      write_str       ; and print it
                    bsr      no_fire         ; wait for no fire button

process             bsr      read_joy        ; read joystick
                    tst.w    fire_button     ; fire button pressed?
                    beq      process_1       ;   no
                    bsr      pause           ; go pause game
process_1           move.l   #player1,a5     ; point to player 1
                    bsr      joy_direct      ; and move it
                    move.l   #player2,a5     ; point to player 2
                    bsr      joy_direct      ; and move it
                    add.w    #1,sound_count  ; increment sound delay
                    cmp.w    #3,sound_count  ; time for sound?
                    blt      process_2       ;   no
                    move.l   #move_noise,-(sp) ; point to move noise
                    move.w   #dosound,-(sp)  ; and output the sound
                    trap     #xbios          ; call xbios
                    addq.l   #6,sp           ; restore stack
                    move.w   #0,sound_count  ; reset delay counter
process_2           bsr      wait_loop       ; delay for skill level
                    tst.w    done            ; any injuries yet?
                    beq      process         ;   no, go loop
                    bsr      find_winner     ; find who died
                    move.w   rounds,d0       ; convert # of rounds to ascii
                    add.w    #'0',d0         ; and compare with current round
                    cmp.b    round_no,d0     ; maximum reached?
                    bgt      new_round       ; no, let's play again
                    move.l   #done_msg,a3    ; point to game over string
                    move.l   #tie_msg,a4     ; default to tie game
                    move.l   #player1,a5     ; point to player 1
                    move.w   score(a5),d0    ; and move score
                    move.l   #player2,a5     ; point to player 2
                    cmp.w    score(a5),d0    ; compare the scores
                    beq      process_4       ; tie game
                    blt      process_3       ; player 2 wins
                    move.l   #win1_msg,a4    ; point to winner 1 string
                    bra      process_4       ;
process_3           move.l   #win2_msg,a4    ; point to winner 2 string
process_4           bsr      no_fire         ; wait for no fire button
                    bsr      flash_msg       ; flash the messages
                    bra      new_game        ; start over

*---------------------

joy_direct          move.w   stick(a5),d0    ; D0 contains joystick status
                    cmp.b    #1,d0           ; is it up?
                    beq      joy_up          ;   yes
                    cmp.b    #2,d0           ; is it down?
```

PROGRAM LISTINGS

Line Attack

by Kirk Stoever

```
                    beq     joy_down        ;   yes
                    cmp.b   #4,d0           ; is it left?
                    beq     joy_left        ;   yes
                    cmp.b   #8,d0           ; is it right?
                    beq     joy_right       ;   yes
joy_direct_x        rts

joy_up              sub.w   #1,yloc(a5)     ; decrement y location
                    bsr     get_pixel       ; get pixel color
                    cmp.w   #0,d0           ; is it black?
                    beq     joy_up_1        ;   yes
                    move.w  kolor(a5),d0    ; add player color to
                    add.w   d0,done         ; DONE flag
joy_up_1            bsr     draw_pixel      ; draw it
                    rts

joy_down            add.w   #1,yloc(a5)     ; increment y location
                    bsr     get_pixel       ; get pixel color
                    cmp.w   #0,d0           ; is it black?
                    beq     joy_down_1      ;   yes
                    move.w  kolor(a5),d0    ; add player color to
                    add.w   d0,done         ; DONE flag
joy_down_1          bsr     draw_pixel      ; draw it
                    rts

joy_left            sub.w   #1,xloc(a5)     ; decrement x location
                    bsr     get_pixel       ; get pixel color
                    cmp.w   #0,d0           ; is it black?
                    beq     joy_left_1      ;   yes
                    move.w  kolor(a5),d0    ; add player color to
                    add.w   d0,done         ; DONE flag
joy_left_1          bsr     draw_pixel      ; draw it
                    rts

joy_right           add.w   #1,xloc(a5)     ; increment x location
                    bsr     get_pixel       ; get pixel color
                    cmp.w   #0,d0           ; is it black?
                    beq     joy_right_1     ;   yes
                    move.w  kolor(a5),d0    ; add player color to
                    add.w   d0,done         ; DONE flag
joy_right_1         bsr     draw_pixel      ; draw it
                    rts

*---------------------

line_A_var          ds.l  1
intin_addr          ds.l  1
ptsin_addr          ds.l  1

mousedata           dc.b  0,0,1,1

save_kbdvbase       ds.l  1
old_joy             ds.l  1
joy_rec             ds.b  3
joy_work            ds.b  3
flagged             ds.w  1
fire_button         ds.w  1
done                ds.w  1
sound_count         ds.w  1
points              ds.w  1
skill               dc.w  5
rounds              dc.w  5

player1             dc.w  0,119,104,0,1     ; joystick,x,y,score,kolor
player2             dc.w  0,200,104,0,2     ; joystick,x,y,score,kolor

old_color           ds.w  3*5
new_color           dc.w  0,0,0             ; black
                    dc.w  1000,1000,1000    ; white
                    dc.w  0,0,1000          ; blue
                    dc.w  0,1000,0          ; green
                    dc.w  1000,0,0          ; red

title_msg           dc.b  27,'H'
                    dc.b  27,'b','1','ONE         '
                    dc.b  27,'b','4','    LINE ATTACK!    '
                    dc.b  27,'b','2',' TWO        '
                    dc.b  0

score_msg           dc.b  27,'Y',32,36,27,'b','1'
score_one           dc.b  '00000'
                    dc.b  27,'Y',32,67,27,'b','2'
score_two           dc.b  '00000'
                    dc.b  0
```

```
round_msg       dc.b  27,'Y',32,42,27,'b','4','       ROUND '
round_no        dc.b  '1       ',0

game_msg        dc.b  27,'Y',32,42,27,'b','4','    LINE ATTACK!     ',0
fire_msg        dc.b  27,'Y',32,42,27,'b','4',' Press FIRE button   ',0
win1_msg        dc.b  27,'Y',32,42,27,'b','1',' ONE is the winner! ',0
win2_msg        dc.b  27,'Y',32,42,27,'b','2',' TWO is the winner! ',0
tie_msg         dc.b  27,'Y',32,42,27,'b','4','      TIE GAME!     ',0
done_msg        dc.b  27,'Y',32,42,27,'b','4','      GAME OVER     ',0

shrink_noise    dc.b  $00,$08,$01,$02,$02,$00,$03,$00,$04,$00,$05,$00
                dc.b  $06,$00,$07,$fe,$08,$10,$09,$00,$0a,$00,$0b,$00
                dc.b  $0c,$00,$0d,$0d,$80,$08,$81,$00,$02,$20,$07,$ff
                dc.b  $ff,$00

explode_noise   dc.b  $00,$00,$01,$00,$02,$00,$03,$00,$04,$00,$05,$00
                dc.b  $06,$1f,$07,$f7,$08,$10,$09,$00,$0a,$00,$0b,$00
                dc.b  $0c,$20,$0d,$00,$ff,$00

move_noise      dc.b  $00,$00,$01,$00,$02,$00,$03,$00,$04,$00,$05,$00
                dc.b  $06,$0a,$07,$f7,$08,$10,$09,$00,$0a,$00,$0b,$80
                dc.b  $0c,$01,$0d,$00,$ff,$00

gr_title        dc.b  'LINE ATTACK!',0
gr_author       dc.b  'by Kirk Stover',0
gr_skill        dc.b  'SKILL LEVEL ',2,' ',1,0
gr_rounds       dc.b  'ROUNDS/GAME ',4,' ',3,0
gr_fire         dc.b  'FIRE to start',0
gr_plus         dc.b  '+',0
gr_minus        dc.b  '-',0

res_alert       dc.b  '[1][Low resolution only!][SORRY]',0

*---------------------

appl_id         ds.w  1
graf_handle     ds.w  1
vdi_handle      ds.w  1

aespb           dc.l  contrl,global,intin,intout,addrin,addrout
vdipb           dc.l  contrl,intin,ptsin,intout,ptsout

contrl          ds.w  0
opcode          ds.w  1
sintin          ds.w  1
sintout         ds.w  1
saddrin         ds.w  1
saddrout        ds.l  1
                ds.w  8

global          ds.w  0
apversion       ds.w  1
apcount         ds.w  1
apid            ds.w  1
apprivate       ds.l  1
apptree         ds.l  1
ap1resv         ds.l  1
ap2resv         ds.l  1
ap3resv         ds.l  1
ap4resv         ds.l  1

intin           ds.w  128
ptsin           ds.w  128
intout          ds.w  128
ptsout          ds.w  128
addrin          ds.w  128
addrout         ds.w  128

*---------------------

                ds.w  512
mystack         ds.w  0
                ds.w  5

                end
```

Line Attack. END

ST USER

BY ARTHUR LEYENBERGER

I use a variety of PCs in my daily routine and get to use just about all of the leading MS-DOS software currently available. For months I have been using Microsoft's *Excel* spreadsheet program for a mailing list application, but between the size of the database and the slowness of windows, I had to try another alternative.

So I decided to use *dBase IV*, the premier database program for MS-DOS machines. The program comes on 14 disks, and the installation procedure took about 45 minutes. I got it up and running, but I was amazed at the size of the program. The whole procedure made me think of the ST, its software and GEM operating system.

Compared to GEM, *Windows* is an incomplete graphical interface. Although it's touted as the latest and greatest in the IBM world, it is not a finished product. For example, to change directories (folders) in GEM, you simply click on the directory name, and the computer goes to that directory and displays the list of files contained in it. *Windows* works the same way. However, to return to the parent directory, *Windows* requires you to type in its name, whereas GEM lets you click on the close button to move you back to preceding directory.

The more I think about this, the more I appreciate GEM. GEM allows you to navigate throughout the directories completely by using the mouse. You only need to use the keyboard for entering the name of new folders or changing the name of a file. In *Windows*, having to use the keyboard for something as simple as returning to the parent directory defeats the purpose for which the graphical interface was designed in the first place.

GEM also makes it easier to copy files. As you know, point, click and drag allows you to copy a file to any other directory or disk drive via the mouse. *Windows* requires that you select the files, choose COPY from a pull-down menu, then type in the destination drive or directory.

These are not the only differences between GEM and *Windows*. I use several of the GEM application programs on a PC such as *Draw* and *Wordchart* and have a good feeling for the time it takes to launch an application program from the GEM desktop. Likewise, I use several *Windows*-based programs such as *Excel* and *Packrat* (a personal information-management system) and can therefore compare the two operating system environments. Let me tell you, the speed of *Windows* is very slow compared to GEM.

Granted, *Windows* is a more sophisticated program, but there is no excuse for the delays it causes both in launching a program and during the operation of a *Windows*-based application program. As I understand it, *Windows* has a lot of redundant code. It is constantly loading segments of code, device drivers, etc., abandoning that code for the next chunk of code, and then loading some of the original code back in again as it needs it. Using *Windows*-based software on anything but a fast AT-clone or 386 machine is an exercise in frustration.

On the other hand, the GEM desktop and GEM-based application programs seem to be more elegantly written, at least as demonstrated by the superior performance of the programs as compared to those that use *Windows*. I'm speculating, but perhaps GEM has more built-in functions, and therefore the software can rely on these rather than have to duplicate the functions themselves. And GEM software doesn't require the latest hardware to run at a reasonable speed.

Another major difference between GEM and *Windows* is their installation procedures. Both use a prompted installation procedure that asks you questions about your hardware setup, including the mouse, graphics card and hard disk. But if you change, say, the type of mouse you are using, *Windows* requires you to complete the entire installation procedure all over again. GEM lets you choose at start-up of the installation process between modifying an existing hardware setup or creating a new one. The GEM method is far more practical, requiring less time to make a change and much less frustration.

Back in the days of CP/M, before the arrival of the IBM PC, MS-DOS and megabyte programs like *dBase IV*, things were a lot simpler. Further, programmers knew that CP/M machines had a maximum of only 64K of memory, and therefore wrote their programs with that in mind. The programs had little overhead and were tightly written and optimized. I remember using *dBase II* on an ATR8000 attached to my Atari 800 and thinking that it was a powerful program.

It's too bad that Digital Research wasn't able to maintain the presence of GEM in the marketplace. GEM got off to an excellent start on MS-DOS machines—a big advertising campaign accompanied the introduction of the product and users were really excited about having such a powerful capability on their PCs. The threatened legal action by Apple Computers caused Digital Research to rewrite the GEM desktop, draining them of time, money and momentum. As a result, Microsoft *Windows* became the dominant force on PCs that it is today.

Unless you routinely use MS-DOS or another graphics environment like *Windows*, you don't appreciate how good the GEM desktop really is. In fact, I admit to taking GEM for granted myself, something I will think twice about now.

GFA BASIC update

MichTron has been around almost since the emergence of the Atari ST. The company is clearly the most prolific distributor of programs for our favorite computer. Some of its programs are okay, others are really good and a few are truly exceptional. One such program is GFA BASIC along with a number of other GFA products that MichTron distributes. However, as of this year, MichTron is no longer licensed to sell or service GFA products. Here's the scoop.

GFA BASIC is a German product from GFA Systemtechnik that has been sold in the United States by MichTron. MichTron paid a percentage of the profits as royalties to GFA and could choose which products they wanted to sell. Apparently, GFA was unhappy because MichTron would not sell *all* of its products in the U.S. and was interested in having its own subsidiaries in the U.S. in order to make more money.

Without describing all of the details about the GFA/MichTron split up, the bottom line is that GFA currently has no U.S. distributor for its products. Further, support for the GFA products will come from GFA itself, in Germany, rather than from MichTron. Upgrades, warranty problems and the like will now be more difficult for the user to get resolved. At least until GFA establishes a new U.S. distributor/publisher. The international phone number for GFA is 011-49-2115-50400.

As for MichTron, they are already starting to sell HiSoft BASIC and eventually the entire HiSoft software line. For more information on HiSoft BASIC, MichTron can be contacted at MichTron, 576 S. Telegraph, Pontiac, MI 48053; (313) 334-5700.

Here we go again

In the early days of my Atari adventure with both the 8-bit machines and later the ST, I was very vocal about copy-protected software. I realized then and still do, the terrible problem of software theft and the need of software manufacturers to make

money on their products. But I always felt that regardless of the specific copy protection techniques used, copy protection interfered with using the programs, especially on a hard disk.

I thought we had seen the last of copy-protected application programs a couple of years ago. The entire industry, not just the world of ST software, abandoned these techniques because they were counter-productive. It seemed that the software thieves were always one step ahead of the manufacturers, being able to crack the protection almost as soon as the new products were released. As a result, copy protection only hurt the legitimate users by being kludgey to use.

Well, just when you thought it was safe to stay in the water, we have a company that is re-introducing copy protection to its software. Migraph, makers of *Easy Draw* among other products, has recently introduced a new product called *Touch-up* that uses a hardware form of copy protection—the "Dongle."

A Dongle is a little gadget that plugs into one of the computer ports. When the software is run, the program checks to see if the Dongle is present before it will operate correctly or at all. In the case of Migraph, the Dongle attaches to the printer port, between the computer and the printer. Simply stated, *Touch-Up* will not run if the Dongle is missing.

There are a couple of problems with "Dongle-mania." First, inserting and removing jacks from the serial and especially parallel ports of a computer should never be done when the machine is on. Doing so risks damaging the internal electronics of the computer and can be costly to repair. Further, the parallel connector on the ST or any computer for that matter has tiny pins that can be bent or broken. Excessive handling of the connectors should be avoided.

Another reason Dongles are a bad idea is that their very use is almost a challenge to some hardware hackers who may try to defeat its use. If someone cracks the protection that the Dongle is intended to provide and the network of software thieves gets hold of the program, the result will be far worse than if no Dongle was used in the first place.

Finally, Dongles or any form of copy protection on productivity software is a nuisance. The inconvenience placed on honest users has never justified their existence. In fact, ill will on the part of the user community is more likely to be the result of protection techniques. Virtually all companies have realized that copy-protection techniques don't work ultimately and that solid documentation, customer support and fair pricing are the keys to making money in the software business. Migraph is taking a step backward with this approach, and I sure hope that it eventually changes its mind.

Litigation wars

It seems that these days, everybody is suing everybody else. If it's not Apple threatening to sue Digital Research over the "look and feel" of the GEM desktop, it's Apple threatening to sue Hewlett-Packard over its *New Wave* graphical interface. If it's not Lotus Development Corp. suing Paperback Software (for its 1-2-3 clone), it's Atari suing the previous management of Federated Stores.

Atari Games (a division of Warner Communications), the arcade Atari, is suing Nintendo under the Sherman Anti-trust Act. Atari Games, which produces titles for the Nintendo Entertainment System (NES) through its Tengen subsidiary is challenging Nintendo's right to include a proprietary chip in each and every licensed Nintendo game cartridge. By requiring this chip in every game for the NES, Nintendo has been able to control how many game cartridges each licensee can sell.

Many companies have been unhappy with the amount of chips Nintendo has supplied them, saying that many more products could be sold if enough chips were available. Nintendo counters by saying that the video-game business crashed and burned in 1984 because there was a glut of inferior products on the market. It intends to not let this situation happen again. At the core of the dispute is the ability of a hardware supplier to control the amount of software available for its system.

Atari's Tengen has announced that it has successfully reverse-engineered the chip and will produce, manufacture and market games that are compatible with the Nintendo game machine. Many companies are watching to see what the outcome of the suit will be. If successful, it could have far-reaching effects on the third-party software market.

Some random thoughts on CES

I've just returned from the winter Consumer Electronics Show (CES) in Las Vegas (well, by the time you read this, that'll have been a couple of months ago). This show marked my 13th visit to the hallowed halls of consumerism, and things have really changed over the years.

In my early years of CES attendance, my attention was drawn to computers, software, video games and the like. Over the years, video games faded from the scene and satellite dishes were the rage. Then satellite dishes were all but forgotten in favor of hot new "home computers" like the Coleco Adam and the Atari 1450XLD.

Video games are hot again; in fact, hotter than ever. So much so, that there was a rumor floating around during the show that it would be renamed to the Nintendo Electronics Show since almost half of one exhibit hall was devoted to Nintendo games. We called it the "Nintendo Village."

This was the same hall that once housed dozens and dozens of computer hardware and software companies, names familiar to any longtime Atari computer user. Unfortunately, exhibits other than relating to Nintendo, Sega and video-game paraphernalia consisted of burping beer mugs, cackling skulls, motorized mail boxes that would slide out to your car, cheap laser-light show imitations, rechargeable batteries, plastic compact disc protectors and audio furniture. Even the few remaining satellite dishes were forced outside due to space limitations.

Biggest disappointment of all was that Atari didn't bother to show up. Not even just with its game machines. For several years there was so much happening relevant to Atari computer users that I would write two separate articles covering the show—one for 8-bit Ataris and one for the ST. This year, it was a video-game CES.

Of all the companies showing new products, two were really exciting. The first I can't yet talk about due to signing a non-disclosure agreement, but you'll be reading about it in a month or two. The other was U-Force.

Briefly, Broderbund's U-Force is a hands-free video-game controller. It's the first video-game controller that eliminates all physical contact between the player and machine. Currently designed only for the Nintendo Entertainment System but soon to be available for Sega and possibly (according to its developers) the Atari, U-Force looks like a large, flat clamshell.

To use it, you move your hands within the three-dimensional range of the right-angle panels, mimicking the actions you would use in real life. Simply describing it does not do it justice; it has to be seen to be appreciated. *Mike Tyson's Punch Out*, was demonstrated; to throw an on-screen punch, you simply jabbed in mid-air. There is nothing to hold, press or wear out. U-Force translates the player's exact motion, velocity and relative position into on-screen action. This thing is neat!

Broderbund said that the U-Force will work with about 40% of existing Nintendo games. It doesn't use batteries, there is nothing to adjust, it just works. Will it be a fad? Will it be around long enough to eventually become available for Atari games and computers? We'll have to wait and see. ■

ATARI®

AUTHORIZED SERVICE CENTER FOR ALL ATARI PRODUCTS

MICROTYME

A DIVISION OF MICRO PERIPHERALS, INC.

P.O. BOX 369 • KETTERING, OHIO 45409

VISA® MasterCard DISCOVER

ATARI®

HARDWARE

ST'S. . .IN STOCK!!!

Color Monitors	CALL
Mono Monitors	CALL
GTS 100 Drive	CALL
SF 314 Drive	CALL
IB 5¼ Drive	199
Navarone Scanner	CALL

PRINTER'S DEVIL HI-RES GDOS FONT & CLIP ART PACKS IN STOCK! (Great for Desktop Publishing!!)

MODEMS

SX-212 300/1200 bps	CALL
Avatex 1200E	79
Supra 2400	139

ATARI ST SCANNERS, SOUND & VIDEO DIGITIZERS In Stock!

!!!UNBELIEVEABLE!!!

HAYES COMPATIBLE 2400 Baud Modem - RS232

$125

$ $ $ SAVE $ $ $

HARD DISK DRIVES FOR ST'S

SUPRA 20 MB HARD DISK

$569

SUPRA 30 MB HARD DISK

$659

SUPRA 60 MB	CALL
ATARI SH 204	CALL

PRINTERS

PANASONIC	**call for latest**
1080/	CALL
1091/ 180 cps	CALL
KX-P110 Ribbon (Blk)	9.95
KX-P Color Ribbons	10.95
STAR	**Call for latest**
NX-1000 NEW!	CALL
NX-1000 Color	CALL
1000 Ribbon (Blk)	6
1000 Ribbon (Color)	8
OLYMPIA	**simply, the best!**
NLQ modes use 18 x 24 matrix!	
NP-30 130 CPS	199
NP-80s 240 CPS changeable font cards	389
NP-136 15 inch	529

ACCESSORIES

ST Dust Covers	from 8
Mouse Mat	9
Power Strip w/Surge	15
Deluxe Power Strip w/Surge	24
TERMINATOR Joystick. . WOW!.	19
EPYX 500 XJ Joystick	17
WICO Ergo Stick Joystick	17
Printer Stand-Heavy Duty	13
Mail Labels 3.5x15/16-500 pk	4
1000 pk	6
PAPER-1000 Shts-Microperf	14
Compuserve Starter Kit	24
On-Line Encyclopedia Kit	36
Printer Cable 6′	19
Modem Cable	17
Supra 64k Printer Buffer	69

MIDI

Midi Cables 5′	6
Software (Hybrid Arts etc.)	CALL

★ ST SOFTWARE ★

10th Frame Bowling	26
221 B Baker Street	28
3D Breakthru	26
3D Helicopter Simulator	34
AB Zoo	21
Advanced OCP Art Studio	31
Air Ball	26
Air Ball Construction Set	17
Algebra 1, 2, 3	ea 14
Aliants	19
All About America	41
Alt	21
Alternate Reality-The City	32
Alternate Reality-The Dungeon	32
America Cooks Series	ea 9
Architectural Design	25
Arctic Fox	26
Art Gallery 1, 2, 3	ea 19
Assem Pro	39
Autoduel	34
Award Maker	27
Balance of Power	34
Bally Hoo	27
Barbarian	26
Bards Tale 1 or 2	ea 34
Base Two	45
Basketball (Two on Two)	26
Battle Droidz	25
Battlezone	19
Beyond Zork	34
Biology 1, 2, 3 or 4	ea 14
Bismarck	28
Black Lamp	17
Blockbuster	27
Boulderdash Construction Kit	17
Brataccas	15
Breach	27
Bridge 5.0	24
Bubble Ghost	24
Bureaucracy	11
Business Tools	26
Cad 3D	65
Captain Blood	33
Carrier Command	33
Certificate Maker	33
Championship Baseball	27
Championship Wrestling	26
Chartpak	34
Chess (Psion)	38
Chessmaster 2000	29
Circuit Maker	54
Clip Art 1, 2, 3, 4, 5, 6	ea 13
Club Backgammon	23
Colonial Conquest	27
Color Computer Eyes	179
Colorburst 3000	25
Compubridge	20
Copyist (DR T)	165
Cosmic Relief	26
Cracked	21
Crazy Cars	25
Cross Town Crazy 8	13
Cyber Control	45
Cyber Paint	58
Cyber VCR	49
Dark Castle	27
Data Manager ST	49
Datatrieve	33
DB Man	159
Death Sword	13
Deep Space	31
Defender of the Crown	33
Degas Elite	39
Desk Cart	69
Diamond Mike	13
Digi Drum	27
Dive Bomber	26
Dr. Drums (DR T)	19
Dr. Keys (DT T)	19
Drafix	129
Dungeon Master	26
Dyna Cadd	449
Easy Draw (Regular)	68
Easy Draw W/Supercharger	99
Easy Tools	33
Empire	38
Expert Opinion	72
EZ Score Plus	99
EZ Track Plus	43
F15 Strike Eagle	26
Fast Basic	67
Fast Basic M Compiler	39
Fire and Forget	26
First Cadd	33
First Letters & Words	34
First Math	27
First Shapes	29
First Word Plus	63
Flash	23
Flash Cache	54
Flight Simulator 2	35
Scenery Disks	ea 18
Font Disks (Pub Part)	ea 20
Fonts and Borders	24
Fontz ST	23
Foundations Waste	26
Fracton Action	26
Frostbyte	17
Gateway	31
Gato	34
Gauntlet	33
Genesis (Molecular Modeler)	59
GFA Basic	39
GFA Basic Book	27

LARGEST SELECTION IN THE U.S.

GFA Companion	34
GFA Compiler	39
GFA Draft Plus	89
GFA Quick Reference Manual	12
Ghosttown	13
Global Commander	28
Goldrunner	26
Goldrunner 2	27
Goldrunner 2 Scenery Disks	ea 7
Gone Fishin'	28
Great Chefs Vol. 1, 2, & 3 Set	39
Gridiron (Football)	19
Guild of Thieves	29
Gunship	26
Hard Disk Backup	23
Hardball	26
Harrier Combat Simulator	34
High Roller	27
Hippo Concept	45
Hollywood Hijinx	19
Home Accountant	34
Human Design Disk	25
Hunt for Red October	34
IB Copy	23
Impossible Mission 2	27
Indiana Jones Temple of Doom	33
Interlink ST	26
International Soccer	26
Into The Eagles Nest	27
Inventory Manager	52
Jet	36
Jinxter	27
Joust	19
Juggler	34
K Resource	36
Karate Kid 2	27
Karateka	23
KCS Level 2	215
KCS-Keyboard Control (DR T)	165
Kid Progs	27
Kids Stuff	27
Kinderama	27
Kings Quest 1, 2 or 3	ea 32
Knickerbockers	12
Label Master Elite	29
Lattice C	109
Leaderboard Dual Pack	17
Leatherneck	27
Leisure Suit Larry	26
Leviathan	11
Liberator	14
Lock On	26
Lords of Conquest	15
Lurking Horror	21
Macro Mouse	25
Magic Sac Plus	115
Magic Sac Roms	CALL
Major Motion	26
Make It Move	47
Marble Madness	27
Mark Williams C	124
CSD Source Debug	46
Master Cad	132
Match Point	25
Mavis Beacon Teaches Typing	28
Megamax C (Laser C)	119
Mercenary	27
Metro Cross	16
Micro Kitchen Companion	26
Microleague Baseball	39
Midi Maze	26
Midi Recording Studio (DR T)	27
Missile Command	19
Mixed Up Mother Goose	21
Modula 2 (Developer's Kit)	99
Moebius	41
Mouse Trap	14
Music Construction Set	35
Music Studio	34
N Vision	29
Neo Desk	20
New Tech Coloring Book	15

★ ST SOFTWARE ★

Ninja	14
Obliterator	27
Ogre	27
Oids	24
Omnires	23
Orbiter	26
Paint Pro	33
Paintworks	14
Paperboy	26
Partner Fonts	21
Partner ST	46
Pawn, The	29
PC Ditto	65
Perfect Match	27
Personal Pascal	66
Phantassie 1, 2 or 3	ea 26
Phasar	59
Pinball Wizard	24
Pirates of the Barbary Coast	17
Planetarium	26
Plutos	21
Police Quest	33
Power Plan	52
Prime Time	27
Print Master Plus	26
Pro Copy	28
Publisher ST	79
Publishing Partner Pro	CALL
Q Ball	21
Quantum Paint Box	31
Quink	11
Read & Rhyme	27
Renegade	14
Road Runner	26
Roadwars	22
Rockford	22
Santa Paravia	19
Scan Art	33
Scruples	29
SDI	34
Shadow	22
Shadowgate	34
Shard of Spring	27
Shuffleboard	19
Silent Service	27
Sinbad	33
Sky Fox	14
Slagon	27
Soko Ban	23
Space Quest 1 or 2	ea 33
Spectrum 512	49
Speed Buggy	29
Speller Bee	29
Spiderman	7
Sprite Factory	26
Spy vs Spy 3 (Arctic Antics)	19
ST Disk Drives Inside & Out	18
ST Gem Programmers Ref Man	15
ST Internals Book	15
ST Intro to Midi Book	15
ST Machine Language Book	15
ST Peeks & Pokes Book	14
ST Pool	21
ST Talk	5
Star Fleet 1	37
Star Raiders	19
Starglider 2	26
Stellar Crusade	36
Stock Market - The Game	18
Strip Poker 2	27
Sub Battle Simulator	26
Sundog	27
Super Base Professional	199
Super Cycle	14
Super Star Ice Hockey	33
Swift Calc St	49
Tanglewood	27
Tau Ceti: Lost Star Colony	11
Temple of Apshai Trilogy	13
Terror Pods	27
Test Drive	27
Three Stooges	34
Thunder	19
Time Bandit	24
Top Gun	11
Trailblazer	33
True Basic	52
Tune Up	34
Turbo ST	36
Typhoon Thompson	23
Ultima 2, 3 or 4	ea 39
Uninvited	34
Universal Item Selector	14
Universal Military Sim.	31
Universe 2	46
Vampires Empire	20
Vegas Craps	24
Vegas Gambler	23
Video Titleing	22
Vip Professional	149
War Ship	39
Wargame Construction Set	24
Winnie The Pooh	16
Winter Challenge	11
Wiz Ball	11
Wizards Crown	26
Word Perfect	239
Word Up	49
Word Writer ST	49
World Games	26
World Karate Championship	19
WWF Microleague Wrestling	33
Xevious	19
Zork Trilogy	46

HOURS: M-F 9 a.m.-9 p.m. EST SAT 9 a.m.-5 p.m.

ALL 50 STATES CALL TOLL FREE 1-800-255-5835

For Order Status or Tech. Info, Call (513) 294-6236

TERMS AND CONDITIONS

• NO EXTRA CHARGES FOR CREDIT CARDS! • We do not bill until we ship • Minimum order $15 • C.O.D. - $3.50 • SHIPPING: Hardware, minimum $4; Software and most accessories, minimum $3 • Next day shipment available at extra charge • We ship to Alaska, Hawaii, Puerto Rico (UPS Blue Label Only), APO and FPO • Canadian orders, actual shipping plus 5%, minimum $5 • Ohio residents add 6% sales tax • Please allow 3 weeks for personal or company checks to clear • All defective products require a return authorization number to be accepted for repair or replacement • No free trials or credit • Returns subject to 15% re-stocking charge • Due to changing market conditions, call toll free for latest price and availability of product. FOR YOUR PROTECTION, WE CHECK ALL CREDIT CARD ORDERS FOR FRAUD.

CIRCLE #103 ON READER SERVICE CARD.





12 Issues $28

$19 OFF THE COVER PRICE

12 Issues with Disk $79

NEW LOWER PRICE

BREAK AWAY FROM THE PACK

The world of ATARI-ST continues to grow by leaps and bounds, and ST-LOG is there every step of the way! We stand apart from the competition by offering more color, comprehensive reviews and in-depth features. SUBSCRIBE NOW!

☐ **12 Issues $28** MCQWW

☐ **12 Issues with Disk $79** DCQWW

☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MASTERCARD

CARD # EXP SIGNATURE

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires July 28, 1989.

the Animation

BY MAURICE MOLYNEAUX

Last issue I covered some of the techniques of conceiving a project. This time we'll talk about how to bring those ideas of yours under control and in a form you can animate.

The Storyboard

The storyboard was originally a tool of animators only, but it's become a standard mechanism for presenting ideas in all types of filmmaking, ranging from live-action shooting to special effects production.

A storyboard is a "shorthand" version of the animation/film you are planning. It should be thought of as a visual outline of the work, sort of a picture-oriented script that uses a series of drawings to indicate, break down and visualize all the action. Looking at a storyboard is a bit like looking at a comic strip, where each drawing represents a particular point of view and/or action. These drawings depict roughly what the finished scenes may look like, including the viewing angle, what appears in view, the size of objects/characters and their relationships with other items in the scene, direction of movement, and so on. It may also indicate drawing style and even colors if you choose to make one that detailed.

Each drawing is usually accompanied by text describing the action, in addition to noting crucial details like the length of the shot, sound and special effects, and so on. Generally, the more complex the action you plan, the more frames in the storyboard. You can storyboard a 20-second pan with one drawing and an arrow indicating the direction of the camera movement. However, a ten-second shot of a character getting off the ground, grabbing a rifle, loading it, aiming and firing, might require between four and six drawings minimum to clearly show all the actions.

Figure 1 (page 38) is an example of a storyboard; in this case it's the infamous Japanimation sequence that was deleted from the *Art & Film Director* video I created for Epyx. Note that a new picture appears every time there is a major movement, change in the scene, or variation in viewpoint. Just by looking at these drawings you can get a feel for how the shot would have looked.

There may not seem to be much to these storyboards, but there is a structure to the way the scenes and actions are presented, which I shall explain now.

As I said last time, the plan for the scene was this: Characters aboard a giant robot spacecraft are conversing about some problem. Momentarily, a kid character in the background starts talking tough, rattling on and on until we can't stand it any more. At this point, Megabit Mouse enters the scene and shoots the kid to shut him up. In animation we must present all this in a manner that will allow the audience to understand where we are, what we are looking at and where the various characters are in relation to one another, while at the same time actually carrying through with whatever business (action) the scene entails.

Let's consider why the storyboards were drawn as they were. The characters are in a giant robot ship heading towards Earth. I established this by first drawing a robot flying towards the Earth (note the arrow indicating how it is to move). I put the moon in the background to reinforce that the planet is Earth. I then reversed angles to view the robot in a close-up, allowing the viewer to see the silhouettes of the characters through the robot's eyes/windows. This not only gives the viewer a better look at the robot, but also establishes its scale, and that it is actually a vessel with people on board.

Next, the viewpoint switches to the interior of the robot. I chose the angle here because it shows both characters (a girl in the background and the hero in the foreground) at the same time and allows us to see out one of the eyes/windows. The viewer would not be confused into thinking that we've shifted scenes to another locale because I've left visual cues to let them know where they are. In the previous angle they saw the silhouettes of two figures through the robot's eyes. In this one they also see two characters, a window shaped like one of the eyes and space outside. Sure, viewers are likely to automatically assume that we are now aboard the robot, but there's no reason not to reinforce this, because it is likely that not all viewers would come to this conclusion immediately, particularly if a lot of sudden scene changing preceded this sequence. The action in this shot is that of the girl asking a question.

We switch to a close-up of the hero's face

stand: STORYBOARDING

as he responds to the girl. This gives us a better look at the hero as he speaks, and it also gives us a different view of the control room, allowing us to see the hitherto unseen obnoxious kid in the background. This change in angle not only adds visual interest, but, more importantly, establishes that the kid is in the room and where he is in relation to the other characters. Thus, when we go to a tight shot of the kid in the next angle, he won't seem to have appeared from nowhere.

The viewpoint switches to the kid, who rattles on and on until Megabit Mouse pops into the scene, levels a gun on the kid and shoots him (in order to shut him up). Unlike the previous angles, where very little changed, this shot has several things happening, and thus it is presented in three sequential drawings. First we see the kid sitting cross armed and complaining. Next he reacts in horror as Megabit Mouse appears and levels a gun on him. Finally, he vanishes in a white flash as the gun is fired (note that Megabit is silhouetted against the flash).

Now, take a few moments and read the text accompanying the storyboard. Notice the description elaborates on the specifics of the action presented. For example, it will not only describe the action of an object or character, but the nature of that movement (in this case, lots of choppy movements and sudden jerks). Accompanying the description of some panels are lines of narration and dialogue, to be spoken by the narrator or the indicated character.

Materials

The term "storyboard" comes from the fact that storyboards, in professional circles, are usually a series of drawings defining the story, which are pinned up in sequence on a corkboard. As the project progresses, sketches are added, deleted, replaced, and the drawings re-arranged on the board to reflect the most recent version. This is still probably the best method of planning your work, making small sketches and tacking them to a large bulletin board. It allows you to easily rearrange and change sections of the storyboard, which is harder to do if you've put more than one drawing per sheet on paper.

If you don't have a bulletin board, you can still draw the pictures on separate slips of paper (unruled index cards are good) that you can easily rearrange and lay out on a table to view.

Some people prefer to storyboard on sheets of paper containing a series of frames already on the page. Many art and professional video-supply shops sell storyboard pads that have frames for you to draw your sketches in and adjoining boxes for writing comments. I never buy these. I use my ST and *DEGAS Elite* to draw a picture containing a series of frame borders (with space for written comments), and dump it to my printer (next time I'll use *Easy Draw* and create an appropriate page to print). I then photocopy a few dozen of them and get to work. You can do whichever method you prefer, though I warn you that the photocopying method will only work well if the copies are made on the kind of paper where ink doesn't smear, pencil lines stick, and the photocopying fluid doesn't rub off and turn the paper gray if you use an eraser.

The sheets I make consist of three rows of three frames each. Many storyboard pads cram dozens of frames onto a single sheet. For my taste these are too small and don't allow enough room for drawing detail (when you need it).

Developing your idea and drawing it

When you set out to start storyboarding, you may wonder just how detailed you should get. There are as many answers to this as there are ideas to be presented and people to come up with them. Insofar as drawing style goes, draw the amount of detail you feel necessary to clearly record what you want. The storyboard example in Figure 1 is more detailed than the boards many animators draw, as they were penciled first and then inked. Pencil sketches alone may be enough for your needs. As to how many drawings are required, generally you'll want to have a storyboard drawing for at least every point-of-view change (like changing scenes, going from a profile to a head-on shot, switching from a long shot to a closeup), and probably for each and every major action in the sequence.

Of course, this is assuming you know in advance all the action and points of view, which is probably not the case. Don't panic. You don't start off drawing all the action and scenes in order, putting the whole thing together in sequence. You start off with whatever you can think of, in any order at all, and sketch it. Once you get a bunch of these together you put them in some kind of order, then study them to see if something new springs to mind or if you can think of a better way to present an action you have drawn.

Whenever you come up with a new idea or a modification of an existing one, sketch it and place it into your fledgling storyboard. Don't throw away any old sketches when you replace or remove them, because you may find at a later date that those ideas rejected at one point may be needed at another point. You may even find that a new idea doesn't work, and you have to go back to the old one. At the very least, the way you presented the idea in a rejected sketch may stir an idea for a shot elsewhere in the project.

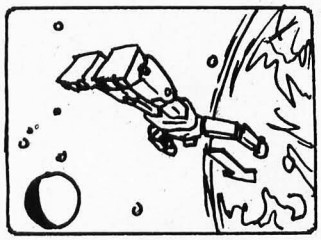
An example is in order here. I'll make up a situation and go through the storyboarding steps.

The story: An ant wanders away from the anthill, but does not get far before some disaster strikes and ends its day (and life) rather abruptly.

Simple, yes? On the surface, perhaps. There's not much we can do with a premise like this—or is there? We could play this realistic, scary, dramatic or even for laughs. How? Consider the following possible refinements of the idea:

Realistic: A red worker ant runs into a black ant. They engage in battle, struggling mightily, thrashing and biting and stinging. But in the end the black ant severs one of the red ant's legs and its abdomen, vanquishing it.

Since this is a realistic approach, the animation would be of real-looking ants,



Text for Panel 1:

—Fade in from black to Space. Moon is to left, Earth looms to right. A typical Japanese robot cruises towards the Earth.

NARRATION: "Film Director gives you a lot of power, and with it you can do better than what is considered 'professional' animation in some circles."

Panel 2:

—CU of robot's face. Eyes are windows, and human figures are silhouetted inside them. Stars drift to imply movement.

Panel 3:

—Cut to: Interior robot. Typical Japanimation types are seen (with two-inch-wide eyes, half-inch-wide mouths and unruly mops of hair). A girl sits a bit back from us, behind the robot's right "eye," while a guy (such as he is) is seen in CU profile. Voices are (naturally) out of synch with the animation and annoying as all get out. The only thing that moves is the girl's mouth and the stars outside the window.

GIRL: "Are you sure we should return to Earth without help?"

Panel 4:

—CU of hero from front. His head turns a bit (a one-frame jerk) from glancing at her to looking straight ahead, and his mouth moves (no blinking!). Obnoxious kid-type sits in BG.

HERO: "Yes, we must do what we can to liberate the people of Earth from the Verbicide! Help or not, we must!"

Panel 5:

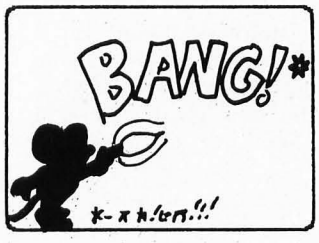
—Medium shot of kid. His big mouth is shown on only two positions: open and closed (no in-betweens), as he rattles off inane dialogue in one of those screechy voices that drive sane men to commit heinous crimes.

KID: "Yeah. We'll teach 'em! We'll show those Verbicidal maniacs who's toughest! Why, I could take them all out, single-handedly. . ." (Etc., etc., etc.)

Panel 6:

—Suddenly, Megabit Mouse appears at screen left, pointing a pistol at the kid. Kid's head jerks (need I have said it?) to face Mega, and he screams.

KID: "Hey! There's a mouse here, and he's got a gun."



Panel 7:

—*Bang!* Screen flashes white as Mega pulls the trigger. Mega is seen in silhouette, and gunflash is also visible. After gunshot, screen fades to black.

NARRATION: "Thank you. As you can see, even a beginner can put more life into a scene than that!"

moving in that rapid, somewhat jerky fashion, that insects move.

Scary: A great shadow falls over the ant. It goes about its business unaware. The shadow pulls back a bit, then a spot of bright light hits the ant. Close-up of the ant starting to twitch nervously, then running about as if in pain. We change angles to look up and a see a shadowy form holding the silhouetted shape of a magnifying glass, a searing point of flaring light at its center. Cut to a medium close-up of the ant twitching. Cut back to a closer view of the flaring light through the magnifying glass. Closer angle on the ant, writhing in pain. More intense view of the light. The ant, light, ant, light, ant, light, until you see the charred corpse of the ant.

Again, a realistic or semirealistic approach could be taken. The use of the shadows, the rapid intercutting between the dying ant and the blazing light through the lens adds the tension.

Dramatic: This is overplayed acting. A family of ants walk along and come face to face with an anteater! The anteater stares them down. The father ant points for his family to hurry back to ant hill. The others react wild-eyed and frightened, but he pushes them away. The anteater watches the family, its tongue flicks. The father takes one last look at his slowly retreating family, then glares steely-eyed at the anteater. He strikes a heroic pose, then marches in a path to draw attention away from his family. He gets eaten as his family reaches the safety of the hill and reacts in horror to his demise.

Because a real ant would not be capable of the expressions needed here, a stylized, more cartoonish type of character is needed, but not funny-looking ones, as we don't want people to laugh at its demise.

Comic: As a cartoon ant walks along, a shadow falls over it. Then, an intense light engulfs it. We see the silhouette of a kid with a magnifying glass focusing light on it. The ant squints at the light, dons a pair of sunglasses and lays down to get a tan. It gets hotter, so it puts on a sombrero, but it catches on fire. It puts up a beach umbrella to make some shade, and it goes up in a puff of black smoke. Sweating profusely, it falls to the ground and drags itself along like a man in the desert. It crawls out of the light, then sits up and wipes its brow, smiling with relief. But then, the light moves back on it, and we see the kid's hand holding the magnifying glass close. The ant looks annoyed, taps its foot, then walks towards the hand and glass, whistling innocently. Suddenly, the ant (instantly sporting a huge set of teeth) jumps and bites the kid's finger.

We hear a loud scream, and the hand and glass zip out of view. The ant turns and walks off, triumphantly brushing its hands together. An instant later, the magnifying glass falls on him, squashing him flat.

In this one the approach is to make the ant do its best to avoid the heat, until it finally gets riled enough to strike back. The humor would come out of its solutions to the scorching heat, and its reactions at the failure. Finally, the surprise punchline comes when it defeats its would-be-assassin's plan to fry it, but is inadvertently squashed by the dropped magnifying glass.

And there are more ways to play it than I've described here. When planning an animation or film you have to consider what sort of feeling you want it to convey and adjust your action and imagery to it. So you'd probably start out scribbling little ideas for the ant, and any other characters or items that might appear, in addition to the staging of a scene, trying different styles and approaches. Figure 2 shows a series of my actual rough sketches depicting various ideas

FIGURE 2

FIGURE 3A

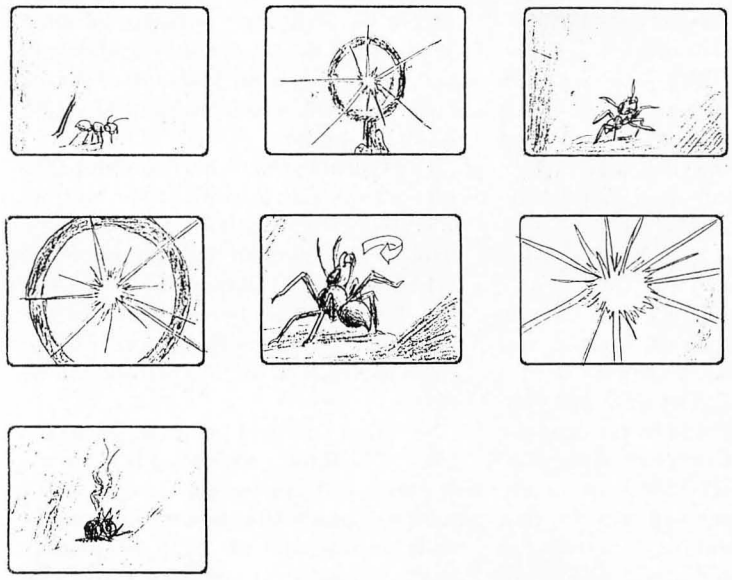
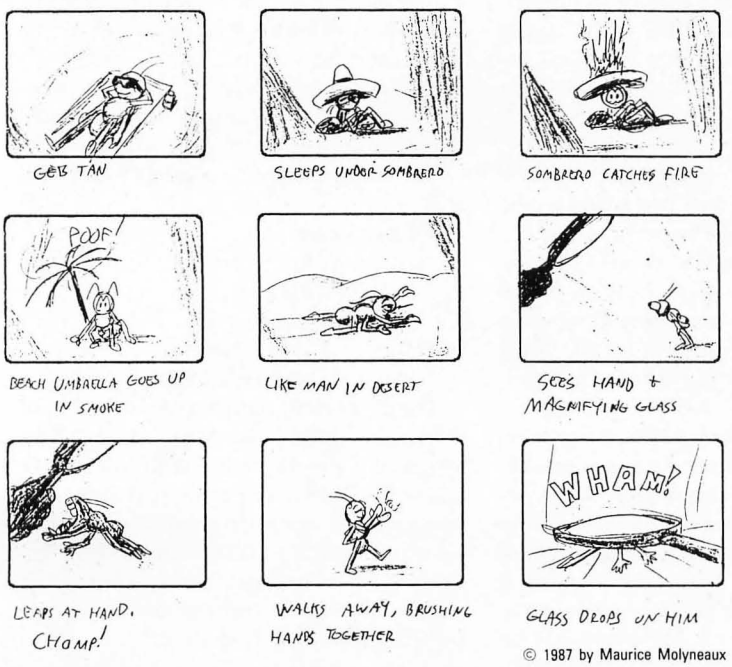


FIGURE 3B



for the ants, anteater, and so on that appear in the various ideas.

Once you've gotten some sketches like this, you'll start narrowing down which ones you like best. Out of the set I drew, I prefer the scary and comic ones, as they seem to have the most "punch" idea-wise. The realistic one isn't interesting to me because I might as well watch a nature film. The dramatic one is too fine a line between melodrama and inadvertent comedy.

However, the stark dramatic intercutting on the scary idea, and the sheer silliness of the comic approach appeal to me, so I'll develop those ideas further. This means some more sketching to get a better idea of each. Here I start roughly storyboarding the action and angles for each, and on this basis start considering which of these two I want to do (I can always change my mind later, but it's best to try to make a good choice right away so to avoid getting to far into something that won't work). Figure 3 contains examples of this, what I call a "preliminary storyboard," which highlight only the most important moments in the action considered.

I again consider the two ideas I've come up with, trying to determine which is the one I want to animate. The dramatic one is neat from a standpoint of visuals, but there's not much plot. By plot I mean a story. You could sum it up as: "Ant is fried by a force it cannot comprehend." Nothing very involved about that. You could tag "So what?" to the description, and that would sum it up. It might be an interesting exercise in cinematic technique, but it's not really much of a film/story.

Next, I turn a baleful eye on the comic approach. What do we have here? We can sum up the plot this way: "Ant finds that things are getting hotter, first it takes advantage by trying to get a tan, but then it tries to escape it through a variety of means. Finally, it realizes that someone is trying to kill it, and it strikes back."

Compare that to the previous idea. See the difference? There is a continuity of action in the latter approach that is absent in the first. The events build up to something. Rather than watching a helpless ant get more and more frantic as it dies, we see a *character* responding to the situation in imaginative and funny ways. It may not seem to be all that much of a difference, but it's an important one; especially in relation to the climax.

Consider that in the dramatic approach the ant's fate is preordained, and it has no control over any part of what happens. In the comic idea the ant's fate is sealed because instead of running away, he strikes back, causing the magnifying glass to be dropped on it. Its action results in a reaction. Action: it bites. Reaction: the magnifying glass drops on it. In this way the ant is a participant in the story, not a passive target.

To be continued

Next time out we'll finish our tutorial on storyboarding, as I develop the ant cartoon into a full-fledged storyboard, and throw in an extra little bonus or two as well.

Blissfully ignorant of the realities of time and space and plain old common sense, Maurice Molyneaux hopes someone will someday discover "retroactive reincarnation" so that when he dies he can come back in a previous life as animation director Chuck Jones. His greatest fear would be to come back as Wile E. Coyote, and in the process have to learn some humility.

Outline Plus

by James Maki

NOTE: Due to the large size of this program, it is available only on this month's disk version or in the databases of the ST-LOG SIG on DELPHI.

Outline Plus is a program that will help just about anyone, including writers, students, journalists or managers, to organize their writing in a more logical, coherent and useful manner. This type of program is often called an idea processor, thought processor or outline processor. It takes much of the tedium out of writing a list or an outline because it automatically takes care of line numbering, subject grouping and format. It is easy to rearrange, add or delete items, so free thinking is encouraged. Up to four windows can be opened and the total number of lines is limited only by the system's memory (approximately 3,000 lines for an unexpanded 520 ST with no desk accessories).

I wrote this program because I am not aware of an outline processor for the Atari ST. It is often included with the more expensive word processors, such as *WordPerfect*, and was announced for the ill-fated *PaperClip Elite* from Batteries Included. I designed the program to compliment any word processor that can import ASCII files.

First and foremost, this program is a creative writing tool. Use *Outline Plus* to bring better organization to your writing, whether it be story plots, articles, a things-to-do list or a student project. Your writing will be better because *Outline Plus* allows both organization and planning to occur before writing begins. Creativity flows unimpeded while random and disjointed thoughts are captured before they are forgotten, to be saved and collected in subject groupings to be reviewed later At the same time, the process is very dynamic because making changes in the outline is so easy.

Brainstorm the main subject groupings and fill out the subtopics and details later as they naturally occur. The multiple windows make it easy to work on several ideas almost simultaneously. When a brilliant flash occurs, switch to another window (file) and record the thought immediately rather than hoping to remember it later. A good idea will never again be lost because *Outline Plus* saves all of your visions; your imagination is free to run wild.

Running the program

Run *Outline Plus* by double-clicking the file OUTLINE.PRG. The program fully supports GEM and uses windows, dialog boxes and alert boxes for all input and output. After loading the program, a GEM selection box will open. Select a file to modify or enter a new filename. *Outline Plus* defaults to an .OLP filename extender.

A new menu bar will be displayed with the titles Desk, File, Block, Output and Menu. All menu bar selections have a keystroke equivalent: The File and Output selections are entered by pressing the control key and the first letter of the command, while the Block functions are entered via the alternate key and the first letter of the command (i.e., Control-O to open a window, Alternate-M to mark a block; see the command summary at the end of the article). Desk will allow access to any desk accessories that were present on your boot disk (Note that the program will not function properly with a desk accessory that utilizes a GEM window that remains open).

Under File are the items Open, Save, Close and Quit. Select Open to retrieve a file for modification or to open a new file. The program will prompt if the file does not yet exist before creating a new file. Select Save to save the current file to disk. Close will close the current window and prompt to save a modified file. Quit exits *Outline Plus*, prompting you to save any modified files.

The Block selections, Mark, Cut, Paste, Clear, and Hide are used to manipulate text blocks between and within windows. All block functions operate on complete lines only! Select Mark to indicate the first line of the block. Move the cursor, if necessary, and select Mark a second time to indicate the last line. The block will be displayed on screen in boldface font. Selecting Mark the second time moves the selected region to the block buffer, ready to paste.

The block region can be removed from the current file by selecting Cut (the block buffer is still available for paste operations). Paste inserts the block buffer to the current file at the current cursor location. Delete clears the block region and deletes the block buffer, while Hide just clears the block from the screen without affecting the block buffer.

The Output selections, Print and Disk refer to the destination of the output. Selecting Print sends the output to the printer. The filename is centered at the top of the page followed by 56 lines of text. The page number is printed at the bottom of the page. No special printer codes are utilized so any printer can be used.

Selecting Disk will print the file to the disk in ASCII form, which can be retrieved by most word processors. This allows the outline to be available in a second window while writing and also can be used to produce fancier output using the output formatter of your word processor.

The Menu selections allow easy access to the multiple edit windows available. The open windows will be displayed by filename and can be made current by selecting the appropriate Menu item or pressing function keys F1 to F4 (corresponding to the window number).

The outline format used by *Outline Plus* is:

I. First Level
  A. Second Level
    1. Third Level
      a. Fourth Level
        (1) Fifth Level
          (a) Sixth Level

The level designators are automatically incremented by the program. Selecting Open will display a saved file or create a new file. When a new file is created, the window will open displaying only a roman numeral I. and the cursor. If information has been previously entered, it will be displayed, and the cursor will be positioned at the first item.

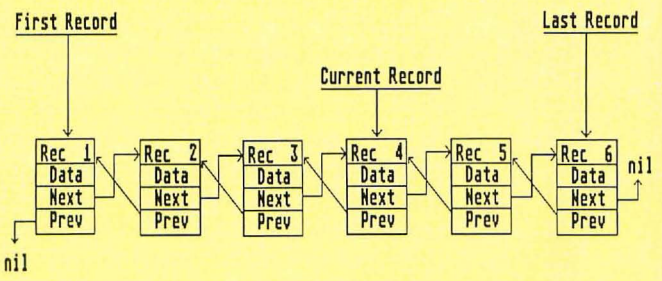


Figure 1. Linked List

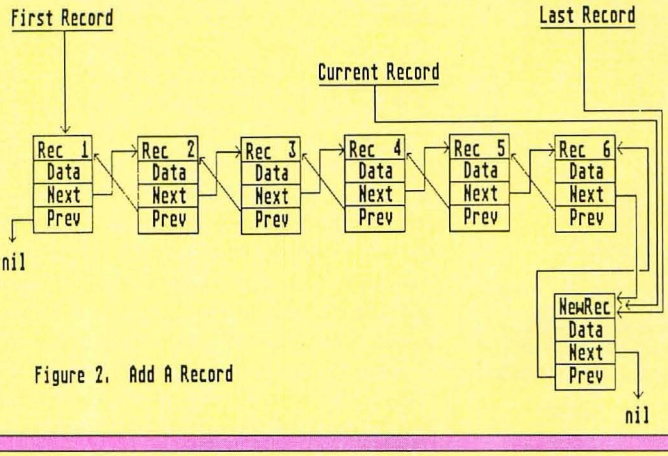


Figure 2. Add A Record

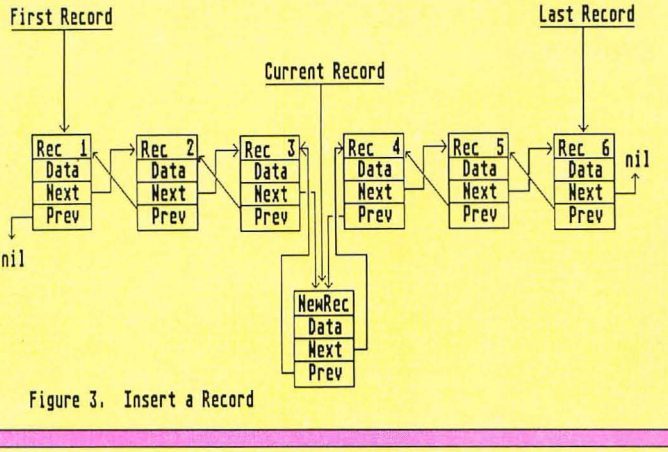


Figure 3. Insert a Record

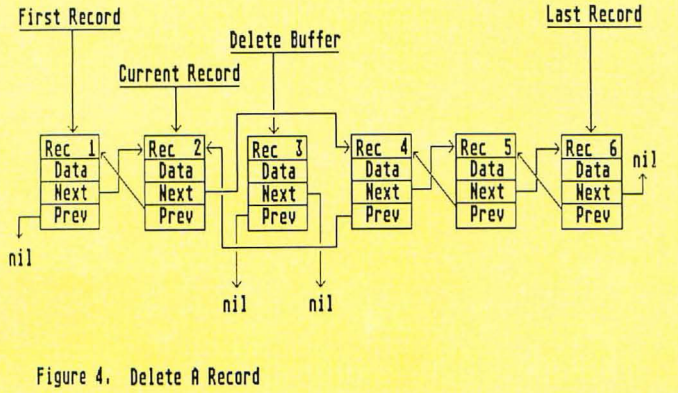


Figure 4. Delete A Record

Command summary

Menu	Keystroke	Action
Open	Ctrl-O	Open new window and file.
Save	Ctrl-S	Save current file to disk.
Close	Ctrl-C	Close current file, prompt to save modified file.
Quit	Ctrl-Q	Close all files and exit program, prompt to save modified file.
Mark	Alt-M	Mark beginning and end of section for other block function.
Cut	Alt-C	Cut marked block. Delete from outline.
Paste	Alt-P	Insert block at current cursor location.
Delete	Alt-D	Delete block and erase block buffer.
Hide	Alt-H	Remove block markers without affecting block buffer.
Print	Ctrl-P	Print outline to printer.
Disk	Ctrl-D	Print outline to disk file in ASCII form.
Window 1 to Window 4	4 F1,F2,F3,F4	Switch between windows. The number of the window appears next to the window title.
	F10	Word search.
	F9	Repeat word search on same word.
	Esc	Clear text from current cursor position.
	Tab	Indent current line of outline.
	Backspace	Erase character to left of cursor position.
	Delete	Delete character at current cursor position.
	Return	Insert new line after current cursor position.
	Insert	Insert new line before current cursor position.
	Clr/Home	Delete current line. Saved in delete buffer.
	Shift-Clr-Home	Clear delete buffer.
	Undo	Insert last deleted line at current cursor position. The delete buffer is a last on, first off stack.
	Shift-Right-Arrow	Same as tab.
	Shift-Left-Arrow	Un-indent line. Opposite of tab.
	Shift-Up-Arrow	Page up.
	Shift-Down-Arrow	Page down.
	Up Arrow	Move cursor up one line, scrolling window if necessary.
	Down Arrow	Move cursor down one line, scrolling window if necessary.

(to page 75)

PRINTOUT AT THE OK CORRAL

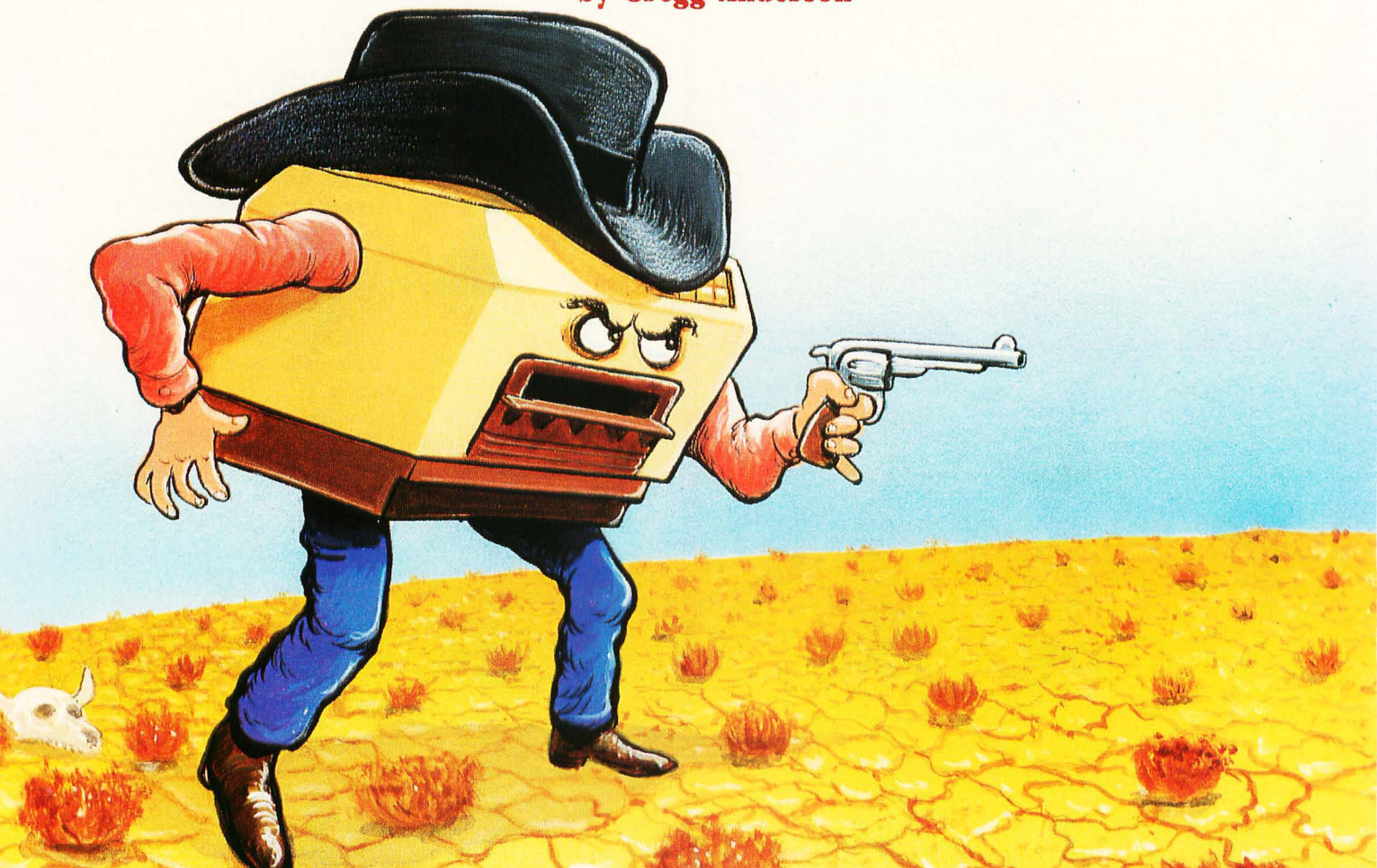
by Gregg Anderson

Over the past two years, a number of really outstanding printers have been released onto an otherwise unsuspecting population. To help that population cope with this flood (and have some fun in the process), I thought I'd do a one-on-one comparison of some of the better (and generally more expensive) printers currently available for ST owners. Since there are so many different types of printers out today, I've had to limit myself to one or two of each basic type. Sorry, no daisy wheels this time around.

In the first ring of this circus is the traditional 9-pin dot-matrix printer, represented by Panasonic's 1092i. The Epson LQ850 will be representing the new, high-speed breed of 24-pin printers. Since laser and inkjet printers are currently all the rage, Rings 2 and 3 are occupied by a couple of them. These include the Hewlett-Packard LaserJet II, the Blaser-Star II, Atari's own SLM804 and the recently released Hewlett-Packard DeskJet.

I chose standard-width printers since (due to their lower costs) these are by far the most popular with most "home buyers." As a rule, wide carriage printers are mainly business machines, though a few have found their way into some home systems.

All printers will be compared on the basis of print speed, print quality and overall cost of printing, both with and without the cost of the printer being included. I'm not including any finance or



depreciation costs in this. The cost factors are simply the price of the ribbon/toner/drum divided by the number of double-spaced, letter-quality pages said consumable is rated to produce. For "price per lifetime" the figure is based on the price per page over the estimated life of the printer, added together with the price of the printer itself.

Be warned that all prices listed here are full retail and rounded up to the nearest dollar. Most printers and their consumables (ribbons, toner, drums, etc.) are readily available at discounts if you're willing to shop around a bit. I'll try and present the advantages and disadvantages of each printer and offer a few thoughts and suggestions on them from time to time.

The programs used for this test were chosen to provide the full spectrum of common computer uses. For text I picked *1st Word* and *Word Writer ST*. For graphics it was *PrintMaster +* and *DEGAS Elite*. Desktop publishing users will find SoftLogic's *Publishing Partner* among the test programs, and for those of us interested in GDOS applications, I included *Supercharged Easy Draw* and the *WordUp* word processor. Please remember that this is a test of printers and not of programs. Don't compare times between the programs since (other than *1st Word* and *Word Writer ST*) all of them are based on different print files and none handle their output in the same fashion.

The Panasonic 1092i

The newest 9-pin from Panasonic is the 1092i, a noticeable upgrade in both speed and quality over the original 1092 with a very affordable price. Offering both Epson and IBM compatibility, this unit is able to handle almost every software package available for the ST. One of the other advantages the Panasonic offers is that its ribbons use a reinkable sponge wheel to save on printing costs.

As expected, the 1092i is one of the slowest printers tested here and uses a dual-pass print for its letter-quality (LQ) mode. While some of its graphics print times are shorter than the DeskJet's or the H-P's, we have to keep in mind that the 1092's overall resolution is less than half these printers. Print time is generally determined by resolution; that is, the higher the resolution, the greater the amount of data that must be transmitted from computer to paper and, thus, the more time it will take.

Though its print quality can't match the laser's or DeskJet's, the 1092i still does an acceptable job in both text and graphics. I was impressed by the solid black printout this unit provides, much darker than the Epson LQ850's print. The most surprising aspect of the 1092i is that it does as well as it does, considering its cost is only a fraction of the competition's.

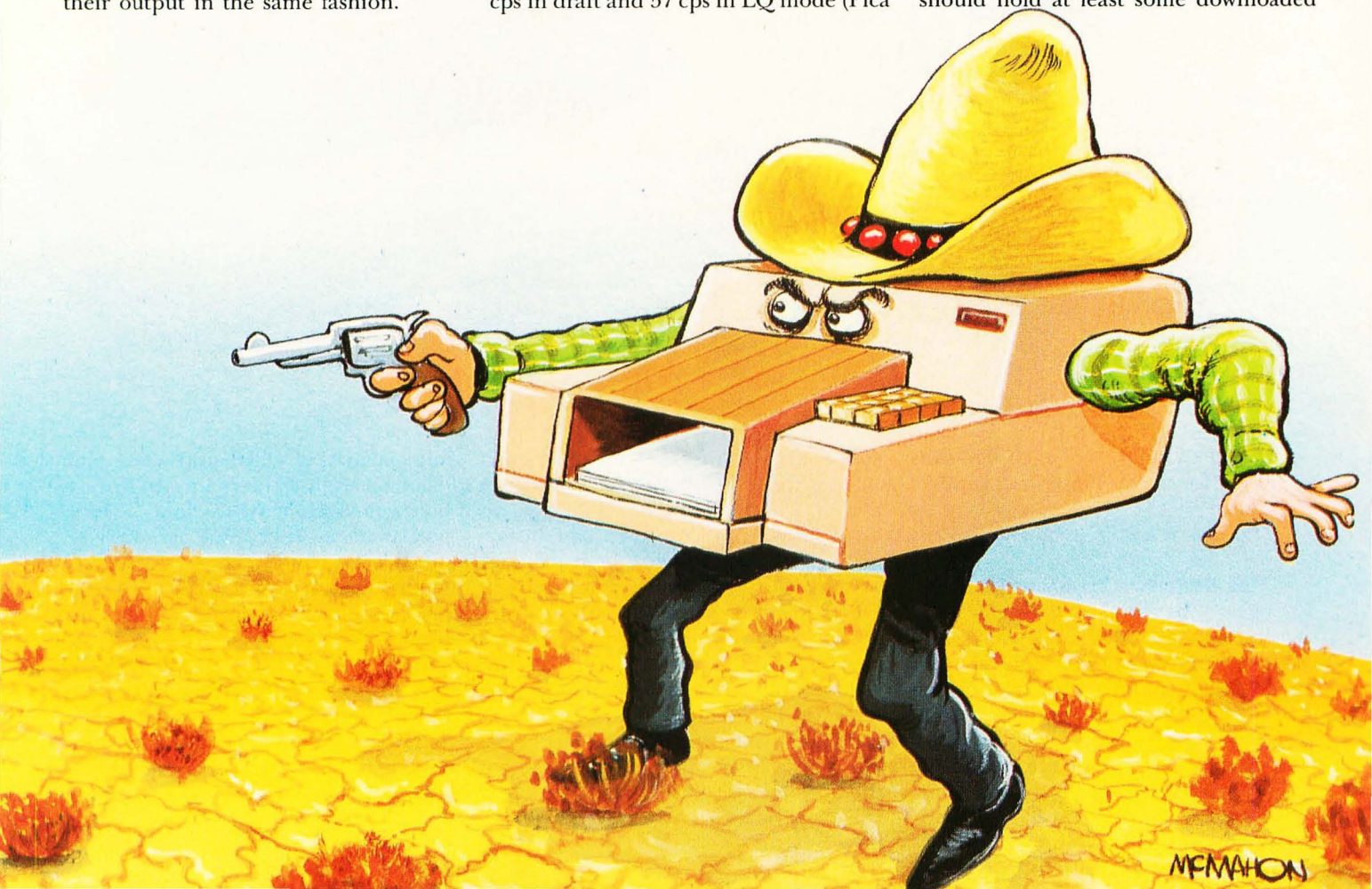
The 1092i is rated by Panasonic at 288 cps in draft and 57 cps in LQ mode (Pica 12 cpi) and offers two draft and two LQ fonts in 10, 12 and 15 cpi. At 6K, the internal buffer is capable of holding a limited number of downloaded fonts. No font cartridge slots are available.

The Panasonic 1092i retails for $500.

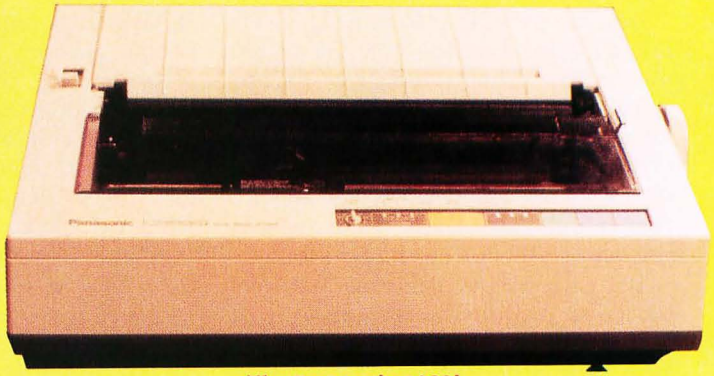
The Epson LQ850

Typical of the new breed of 24-pin printers is Epson's new LQ850. The 850 combines an excellent LQ mode and high print speed with a paper handling system that's in a class by itself. As the LQ850 shares its command set with the earlier LQ1500 system, rather than the LQ2500, it should be supported by most current ST software. For serious graphics users there are GDOS drivers available, and the unit does respond to standard Epson 9-pin ASCII commands for fast, outstanding (single pass) LQ text prints.

I have to admit to being somewhat disappointed by the LQ850's graphics output though. Even with a new ribbon it seemed washed out and gray rather than the expected solid black. The unit also had a tendency to give rather jagged vertical lines, and I suspect my test unit may have been in need of some alignment. The LQ850 is rated by Epson at 264 cps in draft and 88 cps in LQ mode (at 12 cpi). It offers one draft and two LQ fonts in 10 cpi, 12 cpi and condensed prints. At 6K, the expandable internal buffer should hold at least some downloaded

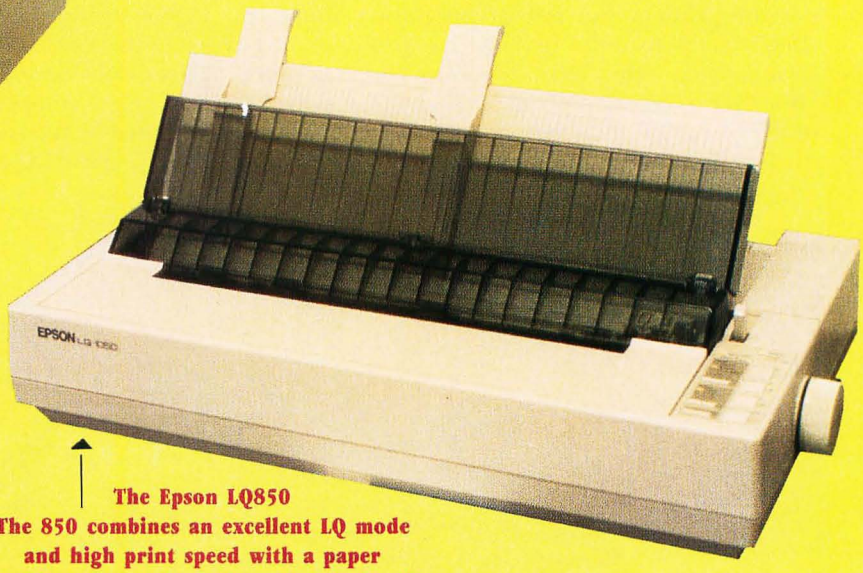


The Atari SLM804
Atari's SLM804 is unusual in that it has no memory or CPU of its own. Instead it relies entirely on the CPU and RAM of the host computer.
The Atari SLM804 retails for $2,000.



The Panasonic 1092i
The newest 9-pin from Panasonic is the 1092i, a noticeable upgrade in both speed and quality over the original 1092 with a very affordable price.
The Panasonic 1092i retails for $500.

The Hewlett-Packard LaserJet II
This very solid printer has sold more units than any other laser on the market today and has become the standard by which all others are measured.
The Hewlett-Packard LaserJet II retails for $2,695.



The Epson LQ850
The 850 combines an excellent LQ mode and high print speed with a paper handling system that's in a class by itself.
The Epson LQ850 retails for $850.

fonts, and there are two cartridge slots available for extra fonts and support boards.

The Epson LQ850 retails for $850.

The Hewlett-Packard LaserJet II

The current standard of laser technology has to be the LaserJet II from Hewlett-Packard. This *very* solid printer has sold more units than any other laser on the market today and has become the standard by which all others are measured. Like most lasers, the H-P offers fantastic quality, near silent operation and a text speed sure to please even the most jaded printer freak. There are some problems though. For one thing the H-P offers no emulations for other printers and thus requires that your software be capable of working with its own PCL command set. Other than GDOS-based software there are few programs out for the ST right now capable of doing that.

A second problem is that, as a rule, lasers tend to be much more expensive to buy and operate than dot-matrix printers, often by a factor of five or more. Something else to consider is that when operating in GDOS you'll require a minimum of one megabyte of computer RAM to hold your various GDOS fonts, with 2.5 megabytes or better being preferred. Adding insult to injury, the printer itself will require at least a full megabyte of internal RAM to handle anything other than simple text. Need I mention that the H-P comes with only 512K of internal RAM?

A final headache is that most lasers, though fast with text and very fast when printing out multiple copies of a single document, can easily take what seems forever to print out that first 300-dpi page

of graphics or GDOS text. This is because the computer must transmit the entire document over the parallel port to the printer before any printing can begin. When you're talking about a megabyte of data, you're talking some time. Of course, once finished the quality is unmatchable, but you'll have to have some patience.

Built into the unit are six fonts in various styles with both standard and landscape (sideway print). If you're not satisfied with the built-in fonts, there are two font cartridge slots and an almost unlimited number of downloadable "soft fonts" available for the H-P. The LaserJet II is rated by H-P at eight pages per minute, but this doesn't include transmission times and is for copies of the document already in RAM.

The Hewlett-Packard LaserJet II retails for $2,695.

The Blaser BlaserStar II

A newcomer to the printer market is Blaser Industries' new BlaserStar II. This unit offers Hewlett-Packard, Diablo and Epson FX compatibility and comes with a full megabyte of memory. Blaser also uses a data compression system that makes the standard one-megabyte work more like 1.5 megabytes, giving enough memory for a *lot* of downloaded fonts or a full 8x14-inch graphics printout. Recently, Blaser Industries also released a lower priced half-megabyte version of this unit for folks interested more in text than graphics.

With its built-in Epson/Diablo compatibility, the BlaserStar doesn't suffer from the lack of ST software support that the H-P (or the SLM804) does. In fact, it should operate with almost all the software currently out for the ST. The BlaserStar II operates at about the same speed as the H-P but retails for around $100 less and comes standard with twice the built-in RAM. One pleasant surprise is that, on a cost-per-page basis, the BlaserStar II costs less than half what the H-P does to operate. Of course, the BlaserStar suffers from most of the same flaws as all the H-P-compatible lasers do when it comes to graphics print speed, but if you can live with that then this unit seems to be the one to beat among the H-P clones.

I did run into an oddity with the BlaserStar II. This unit requires that it be the last item turned on when powering up your system. Turn it on before the computer, and the ST can't access it in any way. Like the LaserJet, the BlaserStar II is a modern, compact design that impresses you with its solid construction and near silent operation. All in all, it's a most impressive unit that not only satisfies the need for emulation but is even slightly faster than the H-P when printing.

The BlaserStar II comes with three fonts, with both italic and bold styles, and offers standard and landscape printouts. Unlike the H-P there is only one cartridge slot available, though the unit will accept downloaded H-P soft fonts. Like the H-P LaserJet II, the BlaserStar II is rated at eight pages per minute under the same circumstances.

The Blaser Industries BlaserStar II retails for $2,595.

The Atari SLM804

Atari's SLM804 is unusual in that it has no memory or CPU of its own. Instead it relies entirely on the CPU and RAM of the host computer (thereby limiting its use to a Mega system or an Atari ST of two megabytes or more). It's also unusual in that it uses the DMA port of the ST rather than the more standard parallel or serial ports. The result of this is that the SLM804/ST/GDOS combination is likely the fastest low-cost laser desktop publishing system on the market today. Graphic printouts of 300 dpi in less than one minute are the rule rather than the exception (an H-P or clone could take up to seven minutes or more to print the same page).

There are some prices to be paid for this speed though. One is compatibility. The SLM804 has had to make do with a less than perfect Diablo 630 emulator for any non-GDOS application, though a recently released revision seems to have cured most of the original bugs. While I did test the Diablo emulator, I was fortunate enough to have a "sneak preview" of the Epson emulator from West Germany's DMC (Design-Marketing-Communication). It seems to work at least as well as the Diablo emulator in text mode and, like the Diablo emulator, the print quality was excellent.

Graphics, however, still need a little work. Though the Epson emulator handles a screen dump beautifully, it has problems with some Epson printer drivers. *PrintMaster +* in particular seems stuck in a "double space" mode, and *Certificate Maker* required at least one megabyte of buffer space to work correctly.

If Atari could make this Epson emulator bug-proof and widely available, it would go long toward making the 804 a good choice as an "only printer." Without such an emulator, the 804 is limited to high-speed GDOS operations and daisy wheel emulations, a frustrating problem for those of us with large, Epson-compatible libraries of text and graphics software. At best, it'll be two years before third-party software houses get around to supporting the SLM804, assuming they ever get around to it.

Unfortunately, due to an apparent lack of interest on the part of SLM804 owners, Atari (USA) has decided not to openly support the Epson emulator. They have instead chosen to encourage software houses to support the SLM804 directly, rather than settle for an emulator. I can understand and respect that decision, but I'd still like to see a workable Epson emulator.

Though the initial price of the SLM804 is fairly reasonable, it has the highest cost-per-page rating of the lasers tested here. Due to the high cost of replacement toner kits ($60) and printer drums ($200), the SLM804 costs almost 4¢ per page to operate. If you include the cost of upgrading your current ST to two megabytes of RAM or the cost of buying a Mega to drive it, you'll find the SLM804 quickly becomes the most expensive printer tested. We need to keep something in mind, however: Even with the cost of a Mega thrown in, the total system cost is still well below what Apple wants for its laser alone.

Though fast, the SLM804 doesn't give you the same feeling of solidness and reliability that the LaserJet and BlaserStar do. Part of this impression is from the fact that the SLM804 isn't as quiet during printing as the others, though the overall noise level is still low. Also the boxy, plastic case lacks the sleek, compact look of the H-P and BlaserStar and doesn't give the same feeling of solidness. So while the SLM804 appears to be well-built, only time will tell if it will match its big brothers in the durability sweepstakes.

The SLM804's font capacity is limited only by the amount of RAM in the host computer and, theoretically, anyway, should be capable of emulating any printer or print language currently on the market. As proof of this, Atari recently announced support for the *UltraScript* print language from Imagen that emulates (and I hear improves on) the rather overpriced PostScript system. Atari's support of this PostScript clone indicates that they want to move the system upscale, with an eye on the Mac/PC-based desktop publishing market and not just the home user.

Rated at eight pages per minute, the SLM804 is the only printer tested capable of doing even a quarter of that speed on single-page printouts. This unit offers the highest overall print speed in its class, with GDOS flexibility. In contrast, it also offers the highest cost-per-page of the lasers tested and limits you to an expanded ST or Mega to drive it.

The Atari SLM804 retails for $2,000.

	Panasonic 1092i		Epson LQ850		HP DeskJet		HP Laserjet II		Atari SLM804		BlaserStar II	
	Time	Quality	Time	Quality	Time	Quality	Time	Quality	Time	Quality	Time	Quality
WordWriterST LQ, 4pp	6:20	C	3:52	B+	3:39e	A	@	@	2:00e 2:25d	W W	1:25e 2:15d	W W
1st Word LQ, 4pp	6:16	C	3:43	B+	3:50h	A	1:54h	W	@e 1:40d	@ W	@e 1:10d 1:10h	@ W W
Degas Elite Screen Dump	1:47	C	1:04	B-	1:58h	B	1:06h	W	:25es :25ds 1:02e	W A+ B	1:42e 1:13h	B W
PrintMaster+ 1pp Graphic	2:14	C+	3:44	E*	4:40e	B+	@	@	@	@	2:15e	A
Publishing Partner, 4pp Text & Graphics	57:44	B	44:40	C+	99:02	A	74:20	A+	46:56d	A	74:58h	A+
EasyDraw 4pp Text & Graphics	20:15	C+	13:52	B+	48:00	A	27:12	W	3:32	A+	26:25h	W
WordUp 4pp Text & Graphics	16:25	C+	11:47	B+	34:44	A	24:42	W	2:35	W	24:31h	W

NOTE: Times and Quality ratings on the Laser & InkJet printers will be followed by a letter indicating what emulation mode or driver the unit was using during that test.
c: Epson
h: Hewlet Packard
d: Diablo
s: Screen Dump (no program printer driver)

@: Software tested has no driver for this printer or the emulator wasn't compatible.

*: Software forces the printer into an 8-Pin Epson FX mode with poor graphics quality.

Times are given in minutes:seconds (IE; 3:45 is 3 minutes and 45 seconds)

Test Conditions:
1) WordWriterST & 1ST Word are based on a 4 page ASCII text file.
2) Degas Elite and PrintMaster+ tests are based on a single page graphic.
3) PubPartner, Easy Draw, & WordUp tests are based on a 4 page text & graphics file with 1 graphic per page.
4) All tests were performed using an Atari Mega4 and Megafile20 Hard-Disk to minimize disk access times.
5) The DeskJet & Lasers were set to 300X300 DPI output, the LQ850 produced 180X180 DPI and the Panasonic produces 120X144 DPI.

All times taken from pressing the final 'print' key (or screen dump) to the finish of the last page. This includes any 'set up' or load time required by the program, but NOT the changing to a special 'outprint' program or a forced 'disk save' such as that used by 1ST Word or Easy Draw.

Quality: W: Wow, A: Outstanding, B: Very Good, C: Good, D: Acceptible, E: Poor, F: No Way

LaserJet tested had 1Meg of RAM
DeskJet tested had 64k of RAM
BlaserStar tested had 1Meg of RAM
Epson & Panasonic had 6k of RAM

	MSRP Printer	RAM Std/Exp	Engine Life Pages*	Ribbon/Toner Drum Life*	Costs; Ribbons Toner-Drum*	Fonts	Price/Page consumables*	Price/Page lifetime*	Warranty Months	LQ PPM	Emulation	Duty Cycle
Pana. 1092i	$500	6k 32k	80,000?	666: ribbon 33k print head	$9 ?????	2LQ 2draft	$0.014	$0.02	24	1	Epson IBM	??????
Epson 850	$850	6k 32k	185,000	1330: ribbon 67k print head	$9 ????	2LQ 2draft	$0.007	$0.017	12	1.8	Epson ASCII only	5,000
HPLJ II	$2695	512K 5Meg	300,000	4,000 toner&drum	$115	6 C,S	$0.029	$0.038	12	8	HP only	5,000
SLM 804	$2000	N/A	300,000	3,000 toner 10,000 drum	$60 $200	NA	$0.039	$0.046	3	8	630/Epson optionial	5,000
Blaser Star II	$2595	1Meg 5Meg	600,000	15,000 toner&drum	$170	3 C, S	$0.0113	$0.016	12	8	Epson 630, HP	10,000
Desk Jet	$995	16k 128k	60,000	400 LQ cartridge	$19	1 C, S	$0.048	$0.065	3	2	HP, Epson opt.	1,000

*: Dot matrix and ink jet printers are capable of both draft and letter quality modes. All listed numbers in blocks with an * indicate Letter Quality numbers. For draft text simply halve the listed costs per page.
Note: Laser printers normally lack a 'draft' mode, the DeskJet is unique in offering both Draft and 300dpi LQ text. An exception are the new printer drivers from Neocept (WordUp), these now offer the choice of draft (1/2 normal) or final print quality on ALL their drivers except for the SLM804.
Though HP lists life expectancy for its ink cartridge at 332 LQ pages (double spaced), Gary Jensen of Dataquest found the unit capable of producing up to 400 such pages in his Research Newsletter for Dataquest. My cost analysis differs from Dataquests' so I've used only my cost figures on ALL tested printers.

Terms Used in this Chart.

MSRP: Manufacture's Suggested Retail Price

Std/Exp: The RAM that comes built into the printer & the Maximum RAM it can be expanded to.

Engine Life: The factory's estimated life span for the print engine (basic mechanics of the printer itself) in double-spaced pages of LQ text. Numbers are based on manufacturer's owner's manuals or conversations with their representitives. NOTE: the engine life of the Panasonic 1092i is an estimate only (no help from owner's manual or Panasonic) and is most likely low.

Ribbon/Toner/Drum Life: The number of double-spaced letter quality pages produced by a new ribbon, toner/toner refill or laser drum, again based on owner's manuals.

Fonts: the # is the number of fonts built-into the printer, C shows it will accept a font Cartridge and S indicates it will accept Software 'downloaded' fonts. Only basic fonts such as Courier & Elite are counted, printers usually include various styles such as italics and bold as 'fonts' as well with lasers adding portrate & landscape as different fonts. I counted only basic fonts to simplify the chart. The SLM804 keeps its fonts in the computer's RAM.

Price/Page-consumables: cost per LQ text page in Toner & Drum, ink, or Ribbon costs. Based on: cost of consumables divided by the # of pages of LQ text produced by that consumable.

Price/Page-lifetime: cost per page of LQ text for the life of the printer, includes cost of printer & ribbons or toner/drum for the claimed life of the unit. Based on: ((cost of consumables divided by the number of pages produced) * the total engine life of the unit) + the retail cost of the unit divided by the engine life of the unit.

LQ PPM: pages per minute of LQ text print in the Letter Quality Mode, as claimed by manufacturer. Graphics print speed will be much slower. This is based on copies of text ALREADY IN THE PRINTER'S RAM!

Emulations: the different printer commands the unit will respond to. Epson: Epson 9-pin, 630: Diablo 630 daisy wheel, HP: Hewlet Packard LaserJet.

Duty Cycle: claimed # of copies per month the unit was designed to produce.

All prices and costs are based on MRSP!!! Discounts are available on most printers and consumable items.
NOTE: As the cost of printing paper is fairly constant, it was NOT included in the 'cost per page' figures.

The Hewlett-Packard DeskJet

The last of our tested systems is the new DeskJet from Hewlett-Packard. This bubble-jet printer is a sample of the newest inkjet technology, and it shows on every page. The DeskJet is an unusual hybrid that combines the advantages (and disadvantages) of both dot-matrix printers and lasers. It offers high-speed draft and LQ modes with a text and graphics print quality that rivals the best lasers on the market today. It also easily matches the speed of most 24-pin printers in text mode and generally exceeds them in quality (300 dpi vs. 180 dpi).

Unfortunately, graphics are quite a bit slower, with up to 30 minutes or more needed for a 300-dpi GDOS page (depending on the GDOS driver). The DeskJet is as quiet as a laser (it barely hums during printing), fast and simple to feed, with ink cartridges and cut-sheet paper installation designed for even the most ham-handed of us. But there is a fly in the soup.

This fly is that the DeskJet is expensive to operate. Though the unit is reasonably priced (under $1,000), it is far from cost effective on a page-per-page basis. This is due to the cost (and 400-page life, based on DataQuest's tests) of the DeskJet's ink cartridges. In addition, the unit is not designed for heavy-duty use. Though fine for most home applications, the rated 50 pages per day output of the DeskJet will restrict it from many businesses. Keep in

mind that this is an estimate only and, like most duty cycles, is conservative.

The DeskJet comes with one font (Draft & LQ) built-in and offers optional Epson emulation ($75), so it should prove reasonably compatible with most ST software. Though it works best with its own custom drivers, the DeskJet will respond to most LaserJet commands, enabling you to start printing right away if you have access to LaserJet drivers (including GDOS drivers).

So the DeskJet offers a set of mixed blessings. It matches most 24-pin dot-matrix printers in speed, offers text and graphics quality to match most lasers, and is priced well below any laser on the market today. In return, you're somewhat limited in the type and size of paper you can use (no labels allowed), and are stuck with higher printing costs, a comparatively short life span, water-soluble ink and having to buy a $75 Epson cartridge that is a must for Atari owners (but at least it's available and it works).

Did you know that copier paper has a right and a wrong side? It does, and it makes a noticeable difference on the DeskJet. Also, while the DeskJet has an optional 128K RAM upgrade board available, it is for downloaded soft fonts only. Adding this board will not affect your print speed or the internal buffer size.

In short, it's up to you to balance the Deskjet's advantages and disadvantages. If you're not planning on doing the printing for a major company or office and want perfect print quality, then this is a printer that must be considered. But if your print needs are heavy and you prefer long-term cost efficiency over short-term savings, then this unit may not be your cup of tea. Keep in mind though, even with the higher cost per page, the difference in price between the DeskJet and a laser will buy a lot of ink cartridges. Something else to remember is that both the DeskJet and the LaserJet come from Hewlett-Packard, an outfit with a proven track record of support and service that most companies only dream about.

The Hewlett-Packard DeskJet retails for $995.

Conclusion

So there you have them: six of the best printers currently available for the Atari ST system. Recommendations? Well, if you're on a really tight budget and your needs don't include high resolution graphics, the Panasonic 1092i may well be your best buy. Its text quality is good and it handles graphics surprisingly well. In addition, the initial cost of the unit is low and, providing you reink the ribbons, its printing costs are low as well.

If you want higher quality print but still want to spend under $1,000, your choice is between the LQ850 or the DeskJet. (I wanted to test an NEC P6 as well, but it just didn't work out.) Though the text quality of the Epson is very good, I have to give the nod to the DeskJet. Despite the higher operating costs and probable shorter life span, the quality of the DeskJet's print is too high to ignore. Assuming you don't mind spending a little more for ink cartridges and take proper care of it, the DeskJet will more than satisfy even the most demanding quality needs of any text- or graphics-oriented computer owner.

And the lasers? Here we have to set some priorities. If reputation and print quality are your prime demands, the H-P LaserJet is the only way to go. It has an outstanding reputation in the industry and a print quality second to none. The only real drawback, other than price, is that it demands that you work within its command set.

If, however, you also want cost effectiveness and enough built-in emulation to handle all your current software, you're going to like the BlaserStar II. This beast matches the LaserJet in just about every category and offers an initial price and economy of use that beats the competition hollow. But be sure to turn it on after the rest of your system, otherwise you're just wasting electricity.

Finally we come to the SLM804. I'd really wanted to see this one lead the pack, but it just wasn't meant to be. Several things hurt the SLM804, with the cost per page and lack of software compatibility being the most important. If Atari could reduce the price it wants for toner and drum replacements (or third-party sources became more common) *and* make a working Epson emulator readily available, it would make the SLM804 a lot more attractive. Also hurting the unit is the fact that you must have at least two megabytes of RAM in your computer to fully utilize it. This limits the potential buyers for the SLM804 to only a fraction of the current ST market, and totally ignores all other computer systems. Still, if print speed and overall print quality are your prime concerns, and you have (or are buying) a Mega system, this is definitely the printer for you.

That's it folks. It's up to you to set your priorities and make your selections. After all, you're the only one who really knows just what you need out of your printer and what you can afford to spend. "But," you say, "you didn't tell us which printer *you* decided was best, the one you bought for yourself." That's right, I didn't. And guess what? I'm not going to tell you, at least not until the next article. Don't you just love a mystery? ▪

A newcomer to the printer market is Blaser Industries' new BlaserStar II.

Gregg Anderson, a captain in the USAF with a background in electronics repair, has been an avid Atari user since 1982. Gregg's roommate, a cat, often leaves messages on DELPHI under his username. "One of these days," Gregg insists, "I'll catch him at it."

A Look at Upcoming Software

by Bill Kunkel and Joyce Worley

The Winter 1989 Consumer Electronics Show proved that there is still blood (as in entertainment software) pumping through the Atari ST's veins. U.S. publishers talk a discouraging line regarding the ST, and virtually no software is developed on it in this country. Nonetheless, there were still some three dozen ST game/entertainment titles either previewed or announced at CES.

Unlike the C64-128 and Apple II—the other systems being treated by software publishers as if they have halitosis—the ST's 16-bit technology allows for easy conversion from Amiga versions. As a result, the ST has ironically reaped spillover benefits from the belated success of its rival.

The main reason for the continued existence of the ST software pipeline is the proliferation of European games into the United States.

But the main reason for the continued existence of the ST software pipeline is the proliferation of European games into the United States. Where once only the very best British and Continental titles made the migration Stateside, European publishers now do a brisk business in U.S. rights. The ST is very popular with European users, as well as a preferred development system for programmers. Many of the titles acquired by U.S. publishers already exist in ST format, therefore, and there is no reason not to release them.

Whatever the reasons, we have 40-plus new ST titles to deal with—from arcade shoot-outs and adventures to sports simulations and classical board and card games—so let's get to it!

Arcade Action

Arcade and arcade-style games continue to be a popular category for the ST. Elite, a leading European publisher, displayed ST versions of Capcom's *Ghosts 'N' Goblins*, Atari's *Paperboy* and Sega's *Space Harrier II*, in addition to its own action game creations. Elite's product is strongly reminiscent of Nintendo's, with an emphasis on side view, horizontally scrolling action games like *Beyond the Ice Palace* and *Thundercats,* based on the animated TV series.

Elite also showed a couple of more sophisticated titles. *Live and Let Die* is a James Bond license that strings together several action games based on sequences from the film, primarily the motorboat chase. *Overlander,* meanwhile, is a fairly slick-looking driving contest that affords the player the ability to earn money and enhance the vehicle grafted on.

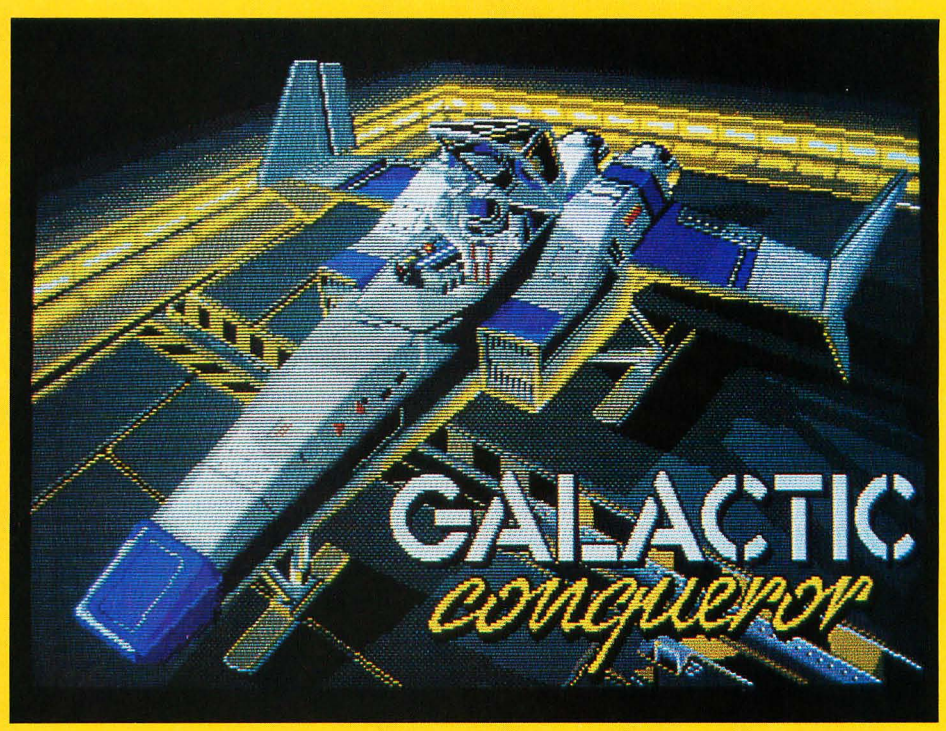
Taito, the longtime Japanese coin-op giant, will be entering the U.S. software fray with top-notch versions of the arcade hits *Alcon, Arkanoid, Bubble Bobble, Operation Wolf, Qix, Rastan, Renegade* and *Sky Shark.* Best of all, each and every title will be released on the ST!

Titus Software, the folks who brought us *Crazy Cars*, will be back with several new action titles. *Galactic Conqueror* is basically *Crazy Cars* in outer space with weapons. *F40 Pursuit Simluator*, meanwhile, is a cross between *Crazy Cars* and *Test Drive* (Accolade), even though it sounds like a flight simulator. And *Titan* is an action-strategy contest with 80 levels of fascinating and difficult puzzles.

Perhaps the hottest new action game for the Atari ST, however, is the visually stunning *Savage.* With graphics so hot they practically singe the screen, *Savage* is actually three games in one: an angled side-perspective scenario with horizontal scrolling, a pseudo-first person game and the final challenge, which combines horizontal and vertical scrolling action. There are traps to avoid, weapons, treasure and magical artifacts to collect, and a first-rate gaming experience to be had by all.

Savage should be available from Rainbird by the time you read this.

Another high-end action game is



GALACTIC CONQUEROR · **From Titus Software**

F-40 PURSUIT SIMULATOR · **From Titus Software**

Japan's *Silpheed,* which will be published in the U.S. by Sierra. *Silpheed* places the user inside a super dogfighter with which he must defend the good space empire from the evil space empire. *Silpheed* boasts 20 levels, 30 different enemy ships and nine original musical compositions.

An action game of a different color is *Ringling Bros. Barnum & Bailey Circus Games* from Keypunch. Designed by Subway Software (Arnie Katz, Joyce Worley and Bill Kunkel) and programmed by England's Tynesoft, *Circus Games* allows the user to walk a tightrope, tame tigers and try his/her luck at horse tricks amidst the clowns and sawdust.

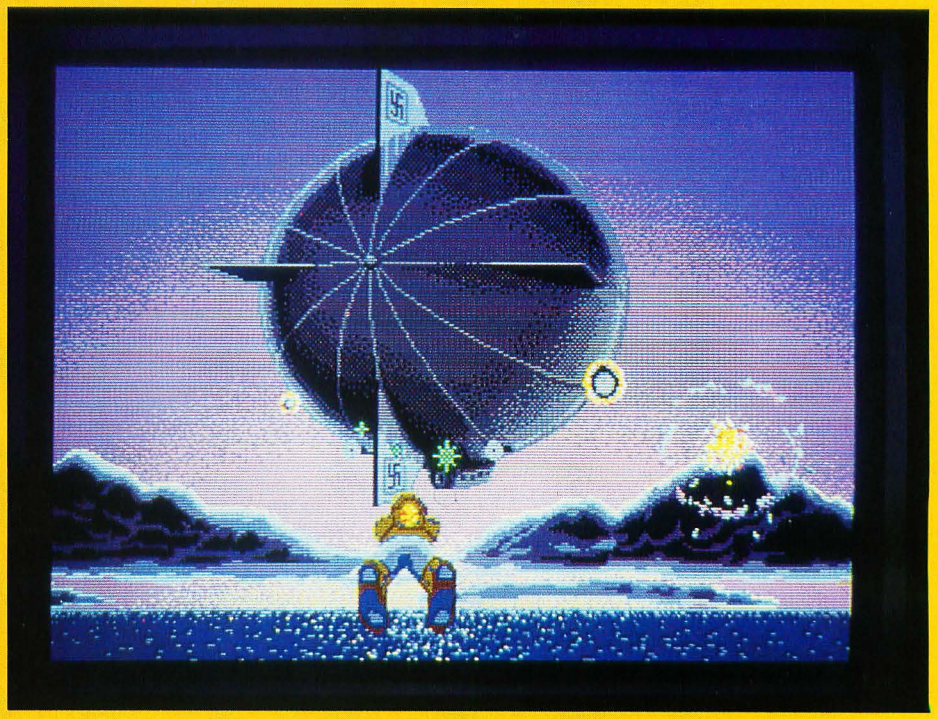
Battles Great and Small

Fans of combat, on both a grand and mano-a-mano level, have several choice morsels, as well as one seven-course meal, to look forward to. On the choice morsel side, Mindscape will be releasing Cobra Soft and Infogrames' *Combat Course*, an innovative military training simulation with four built-in courses and a construction set option that allows users to roll their own. The playscreen is broken into three horizontal strips with the bottom level serving as command console. Each of the top two strips is subdivided into four windows. This modular set-up allows the program—and, eventually, the user—to adapt to a particular scenario.

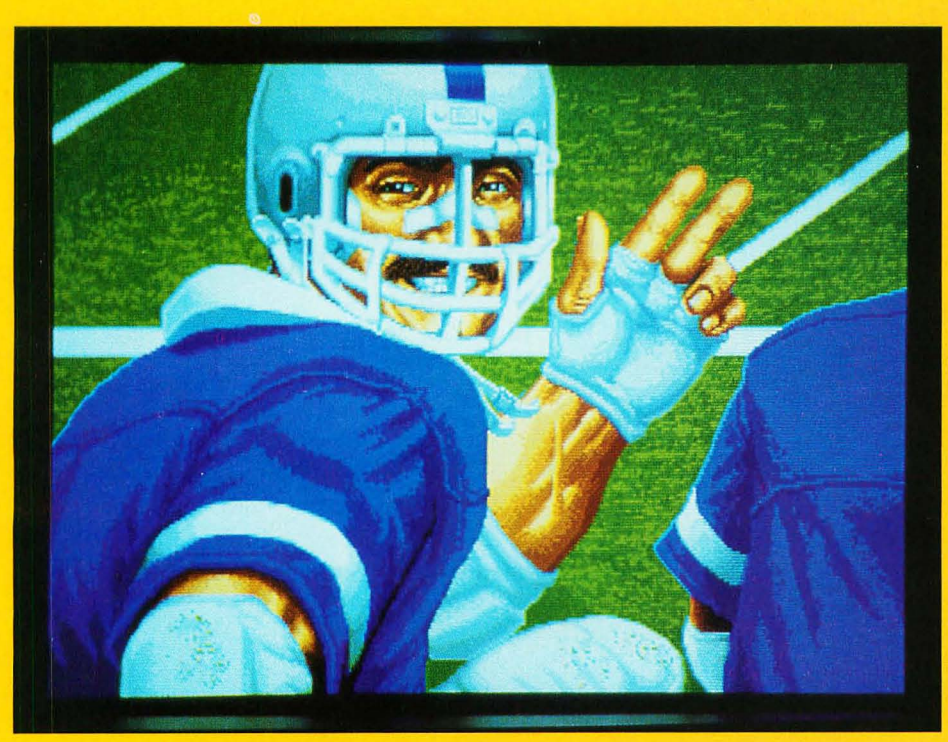
Another tasty tidbit, especially for fans of one-on-one combat, comes from Microprose's new Medalist International label. *Spider Man and Captain America in Doctor Doom's Revenge* by Paragon Software is not only the year's longest title but the only one to feature as many as three licensed characters in it. The program alternates beween noninteractive comic-book pages and one-player side-view fight scenes between various Marvel Comics hero and villain types.

The comic-book material reads more like a Marvel stamp collection than an actual comic, with one or more new characters introduced in every panel—including several generic superguys from Paragon's own fevered imaginations. The fights are mostly standard karate stuff, with some customization. Spidey can web

LORDS OF THE RISING SUN · **From Cinemaware**



ROCKET RANGER · **From Cinemaware**



TV SPORTS: FOOTBALL · **From Cinemaware**

THE HIDDEN DIAMOND CAPER · **From Epyx**

sling, for example, but it isn't really well-integrated into the game play. Cap, on the other hand, gets to use his familiar shield as a defensive weapon.

The seven-course meal referred to above is the latest entry from Cinemaware: *Lords of the Rising Sun.* In what looks to be Cinemaware's finest work yet, *Lords* combines the sweeping grandeur of a "cast of thousands" movie with a series of spectacular action and action-strategy minigames. The setting is 12th century Japan, and the subject is a family feud over the loyalty of the samurai armies that control the Nippon Empire.

Lords boasts extraordinary graphics that evoke both the delicate Oriental beauty and the barbaric cruelty of that era. The action games include a scenario in which the player, armed with a sword, must deflect and redirect shuriken thrown by an advancing ninja. But the most singular and innovative feature of the games is the battlefield scenario. Employing a "magic wand" as a cursor, users command their sprawling troops over the field of combat. If successful, the victorious general then mounts a horse and pursues the defeated and fast-retreating warlord through the nearby woods.

Cinemaware will also be releasing ST versions of several of its hottest titles, including *Rocket Ranger,* an eye-popping art deco tribute to Commander Cody and the cliff-hanger serials of the '40s. Also scheduled for ST release are *The Three Stooges* and *TV Sports: Football.*

Of course, it wouldn't be a software season without a new karate title, and Mastertronic's Arcadia action line fills the bill with an ST version of the Trade West arcade hit, *Double Dragon.* In this popular variation on the traditional martial arts game, players move through a variety of urban and arboreal settings while battling an army of kung fu fighters.

No Time for Sports?

The sports category is shockingly thin this year, with a paltry three new titles announced. Fortunately, all three look promising. Cinemaware leads the way with its first sports entry, *TV Sports: Football,* a detailed and highly realistic simu-

lation that features all the slickness and glitter of a network NFL broadcast. *TV Sports: Football* can be played as either an action or stat contest with graphics the likes of which have never been seen on a computerized gridiron.

Mastertronic offers the European version of "football" in *World Trophy Soccer*, a visually impressive action contest with slick scrolling and excellent animations. And Epyx explores the world of future sports in *Skate Wars*, billed as "a thrilling and violent cross between hockey, soccer and outright war."

The World of Adventure

There is certainly no shortage of adventure games, on the other hand, from role-playing to action-adventures and everything in between. Sierra has several ST offerings in this area, including the latest installments in the various "Quest" series.

King's Quest IV—The Perils of Rosella casts the player as Rosella, daughter of the king who must make a perilous journey to obtain the sole magical item that can save her father from death. In *Space Quest II—The Pirates of Pestulon*, interplanetary nonentity Roger Wilco once again braves the nerdish and comical adventures created for him by Mark Crowe and Scott Murphy, known collectively as the Two Guys from Andromeda.

Finally, a surprise: an adventure entry from Sierra that doesn't have a roman numeral in the title! *Gold Rush* by Ken and Doug MacNeill allows the user to embark on any of three great scenarios set during the great gold rush. In one, the player treks to Central America; in the second, they cross the American frontier; while the third expedition takes the watery route, heading from the eastern United States down to the tip of Cape Horn and back around to California.

Also due from Sierra later this year are *The Plantation Murders, Code Name: Ice Man* and *King Arthur. The Plantation Murders*, co-authored by Roberta Williams and novelist Jackie Austin, will be an attempt to blend slapstick and suspense in a murder mystery set in a 1920s New Orleans plantation. *Code Name: Ice Man* is a Middle East political thriller written by former Navy SEAL Johnny Westland. *King Arthur*, meanwhile, will attempt to tell the saga of Camelot and the Knights of the Round Table in the Sierra animated adventure format.

SEARCH FOR THE TITANIC · **From Capstone**

Role-playing fanciers will want to check out the first game in a new series from SSI, *Demon's Winter.* In this game, the player mounts a party and expedition to defeat the demon-god Malifon, a dude so powerful that, even while trapped inside a volcano, he was able to cast the entire world into a deep freeze.

Fans of fantasy will likely be even more excited by the prospect of Mastertronic's impressive epic, *War in Middle Earth*, based on the writings of J.R.R. Tolkien. This epic contest combines elements of fantasy role-playing, war gaming and the animated adventure as the player strives to bring the great ring to the Crack of Doom in Mount Thunder, opposed every step of the way by Sauron and his minions. *War in Middle Earth* offers a 36-screen scrolling map of Middle Earth containing thousands of locations with more than 80 animated characters and armies. Gamers who enjoy Icom adventures (*Deja Vu*, *Shadowgate*, *The Uninvited*) will want to keep a private eye on the retail shelves for *Deja Vu II: Lost in Las Vegas.* In this adventure, the player is abducted by a pair of Chicago thugs, beaten unconscious and awakens in the bathroom of a cheap Vegas hotel—and that's all before the game even starts! *Deja Vu II* uses the standard Icom interface that eliminates the need to guess at commands since all options are available via menus.

An adventure game of a somewhat different type is represented by *Devon Aire in The Hidden Diamond Caper* (Epyx), which uses a unique angled overhead perspective to present a series of physical puzzles. The player, as Devon, a reformed felon with educated fingers, is hired by the Widow Crutchfield to find her priceless jewel collection. It seems her late husband, world-class eccentric Montague Crutchfield, stashed the jewels in secret spots

throughout the labyrinth-like Crutchfield Manor. The player manipulates Devon about the various rooms, rendered in a pseudo 3-D style vaguely reminiscent of the old Atari coin-op, *Crystal Castles*.

But the most off-beat adventure of all is Capstone's *Search for the Titantic*, a multistage simulation in which the user hones his or her oceanographic skills by locating the exploring undersea wrecks, then heads off in search of the most famous sunken ship of all time. *Search for the Titanic* features a minimum of 75 wrecks, a choice of four ships and more than 100 crew members, a variety of authentic equipment, 47 ports of call, over 100 navigational charts and enough detail to consume a computer-bound treasure hunter for hours.

Recreating the Classics

A small but impressive selection of classic board and card games will be available to the ST user in 1989. Sierra will offer *Hoyle's Book of Cards,* an Omnibus collection that features such card game favorites as Solitaire, Old Maid, Hearts, Cribbage and Crazy Eights.

Board game aficionados, meanwhile, will want to investigate the excellent ST versions of *Risk* and *Clue Master Detective. Risk,* the classic strategy game from Parker Bros and *Clue Master Detective,* a recent revamp of the Parker Bros classic *Clue*, are both scheduled for a spring release by Mastertronic's Leisure Genius Line.

Summing Up

As the saying goes, where there's life, there's hope. And in the computer business, as long as there's software, there's life. As long as the pipeline contains product, the ST will be an excellent choice for the home computer user. ▪

The Latest in Game Controllers

In case you hadn't noticed, joysticks are back.

Not that they ever really went entirely away, of course, but joystick makers suffered mightily during the great video-game shakeout of the mid-'80s, and only recently began to stick their heads out of the bunker. What they found upon emerging was not the burnt-out husk they feared, but a world bustling with computer users looking for replacement controllers. Moreover, video games had not only made a comeback, but looked stronger than ever.

So, after pinching themselves to make sure they weren't dreaming, they dusted off the inventory, fished the R&D plans out of the wastepaper basket and plugged the assembly line back in. Joysticks were once more on the march!

Look, Ma, no wires!

Leading the way in this new joystick revolution has been Camerica and the Freedom Stick, the first successfully marketed remote-control joystick. Camerica's new Atari/Commodore wired sticks are the Magnum (bubble switches), Warrior (bubble switches) and Micro Master (superior microswitches).

Users who like bright colors and suction-base joysticks will get off on Beeshu's line, which includes the Atari-style Hot Stuff, a grip-handle joystick that comes in hot pink, kelly green, bright yellow and sky blue with two sets of action buttons.

Then there's the Advanced Gravis Joystick Controller, a basic black, pistol grip, handle stick with eight-position centering tension control, three independent microswitch action buttons and a one-year warranty.

The Cadillac of joysticks, however, continues to be the Kraft, the most elegant, durable and adaptable joystick ever made. Kraft makes two different Atari-compatible joysticks, and even the less expensive model is well made and extremely comfortable.

Hey, you—stick!

The heavy interest in controllers right now, however, centers around a pair of highly experimental controllers from Broderbund and Mattel.

Mattel showed a revolutionary new "glove"-style controller at the winter CES that analyzes kinetic action on the part of the wearer and translates it into standard control signals.

Broderbund, meanwhile, offered the U-FORCE, which might best be described as an invisible joystick. The U-FORCE consists of a plastic case that opens to resemble a small laptop, but instead of a LCD screen, there is a black reflective surface. And instead of a keyboard, there is a similar surface with a hole and several buttons. The player "pretends" to grip the invisible stick and the movements of the user's hand(s) are translated by the U-FORCE into joystick-type commands.

Joysticks of the future?

There seems to be no consensus on whether or not the joystick will remain the primary control device for entertainment software, but past attempts to move users "beyond" the familiar device have invariably crashed in flames. Remember Le Stick? How about the headband controller that helped drive Synapse out of business?

The self-centering joystick possesses a familiarity that seems overwhelmingly desireable to most users and no technological advance has yet persuaded them to abandon it. ▪

—Kunkel and Worley

• *Arnie Katz, Bill Kunkel and Joyce Worley are synonymous with computer entertainment journalism. The trio has covered electronic games for leading publications.*

IAN'S QUEST

Ian Chadwick

I was having a coffee with some fellow ST enthusiasts the other day and one of them asked me, "What language do you program in?"

"BASIC," I said.

The other looked at me aghast. One even gagged, "*BASIC*? You're not serious!"

"I am," I replied evenly.

"But, but . . ." he spluttered, "Why not C? *Everyone* programs in C."

"I don't."

Another one banged his fist on the table and declared, "Pascal. That's the high-level language to use. It's easier than C."

"It's not easier than BASIC," I retorted mildly. "BASIC is about the easiest language I know."

"What about speed? BASIC is so *slow*."

I shrugged. "Not if you use a compiler."

"But what about GEM calls? VDI? AES? BIOS? XBIOS? ST BASIC can't do any of them without some heavy poking. You might as well write in assembly if you're going to do everything that way."

High-level programmers generally agree that assembly language programmers are really nocturnal aliens who think in hexadecimal.

"You can get a lot of direct access to system calls in GFA BASIC 2. In Version 3, you can access all of them directly, the same way you can in C or Pascal, *and* it's a structured language," I said.

Anyway, the conversation continued in that vein. No one asked me if there was, in fact, a compiler available for GFA BASIC 3 (there isn't, at least not in early January when I wrote this, and I'm not going to phone Germany to find out right now).

Anyway, I *like* BASIC, and I refuse to apologize for the language. I am particularly fond of interpreted languages. Most languages like C, Pascal, Modula 2 and some variations of BASIC have no inherent interpreter. The process of writing and debugging code with them goes something like this:

1) Start the text editor. Write your code. Save same. Exit the text editor.

2) Start the compiler. Compile the code (sometimes this is a two- or three-step process involving linkers and multiple-pass compilation), and wait until it's finished (a period anywhere from a minute to forever). Exit the compiler.

3) Run the new program. Write down any problems, changes or bugs on a piece of paper. Exit the program.

4) Start the text editor again. Load the source code. Look at the piece of paper. Try to rewrite the code from your notes. Save the code. Exit the text editor.

5) Start the compiler and go through the whole process again, watching the hard-disk light blink on and off. Try to think up a song that matches the rhythm. Exit the compiler.

6) Run the new program and write down any problems, changes or bugs on another piece of paper. Exit the program.

7) Start the text editor again. Load the source code. Look at the new piece of paper. Try to decipher what EXC.IN MNU COL 2 means. Rewrite the code from your notes as best as possible. Save the code. Exit the text editor.

8) Start the compiler. Consider using this waiting time to finish reading *War and Peace*, or take a night school course. Tap your fingers on the desk. Look up the word "glacial" in the dictionary. Exit the compiler.

9) Run the program again and wonder how you managed to miss another great gaping hole in the code. Scribble furiously any problems, changes or bugs on yet another piece of paper. Exit the program.

10) Start the text editor and load the source code. Think seriously about taking a handwriting course. Try and remember what was so important about the "screen shrink" that caused you to write it in big letters all over the page. Edit and rewrite the code from your notes, and hope you got it right this time. Save the code. Exit the text editor.

11) Start the compiler. Give serious thought to throttling the computer dealer who promised you this was the fastest compiler on the market. Then wonder what the guy using the slowest one is thinking right now. Consider a new career in shoe sales. Exit the compiler.

12) Run the program. Scream and shout and curse about the one stupid mistake where you forgot a matching quotation mark, and watch the program spew a series of bombs on the screen. Tear several pages off the pad, and write savage remarks about the code, the origin of the language, the stupidity of programming and the need for artificially intelligent text editors. Reboot the system, and try again.

13) Start the text editor, and load the source code. Look at your notes and wonder how and when they got changed from English to Babylonian script. Try and remember which piece of paper is the most recent. Try and remember if the change you're making is really correcting the problem, or if you're just undoing the last change you made and writing the code back the way it was 17 compiles ago, when it didn't work then either. Try to remember why you wrote "ghj shgs jahsf!!!!" and what it means. It's probably very important. Exit the text editor in frustration.

14) Start the compiler. Watch the error messages scroll by too fast to note on the wads of paper covering your desk. It's the only fast thing in the entire process. Ignore them, and compile anyway. Wait. Wait. Wait. Practice, patience. Chant your mantra. Wait some more. Exit the compiler.

15) Run the program. Smile and clap your hands as everything seems to work properly. Then watch in dismay as a simple file I/O turns into an unexpected hard-disk format. Search frantically for more paper to make little notes as the program runs itself rapidly into oblivion and erases weeks of hard work. Write madly. Weep openly.

Get the picture? Me, I like to test and hack in the interpreter before I get to the compiler stage. That way, I am reasonably assured of a stable program, fewer pieces of paper, a lot less hair pulling and considerably less time spent watching my hard-disk light flash.

It's no secret that I'm a fan of GFA BASIC 2.x (and will be of Version 3.x when bugs are ironed out and when a compiler makes the scene). I'm leery of GFA, now that it no longer (as of this writing) has a North American support and service group, MichTron having been dropped under circumstances that appear less than savory. How is it going to be able to update all the current 3.x owners with a non-buggy version?

MichTron is not asleep at the wheel, however. It is bringing out a new BASIC—one very popular in U.K., Highsoft BASIC. It doesn't have an interpreter. That worries me. I've been told online that the system is so constructed that you can go from text-editor to compiler to running the program

and back to text editor seamlessly, without any significant pauses. That may alleviate the syndrome described above; we'll see in a future review.

In the meantime, another U.K. product has made it to our shores in the guise of *STOS* "The Game Creator," from Mandarin Software, distributed by Terrific Software out of San Francisco.

STOS bills itself as a "revolutionary package that has everything you need to create fast, exciting games quickly and easily." I'd hesitate to call it revolutionary, but I'll agree with the rest. *STOS* is really BASIC: line numbers, spaghetti code, the whole bit. But it has more than 300 commands with many game-specific functions not found (as such) in any other language I know of.

For example, you can create sprites easily, display up to 15 on the screen, move them, animate them, change their colors and so on—all easily, with fairly direct commands. *STOS* provides not only a decent sprite editor, but simple means to cut sprites out of *DEGAS* and *Neochrome* pics, or to copy them from another program. Take a look at these lines:

```
SPRITE 1,10,100,2
MOVE X 1,"320(2,-6,0)L": MOVE ON
```

This translates as: Put Sprite #1 from Bank 2 on the screen at Location 10,100. Move Sprite #1 in the X axis, starting from Location 320, at Speed 2 (fast), six pixels to the left each time (a positive number moves it to the right), indefinitely (a number other than 0 is the iteration count, the number of times the step is repeated). L is for loop and means continue this instruction from the start once done (the sprite moves off the screen). MOVE ON activates the command. Pretty straightforward. Then add this line:

```
ANIM 1, "(2,5)(3,5)(4,5)(5,5)L": ANIM ON
```

This is the animation sequence: Sprite 1, Images 2 through 5, are each displayed in sequence for a duration of 5/50 of a second. L is the loop and ANIM ON activates it. In two lines, we've placed, moved and animated a sprite! This simplicity and ease for otherwise difficult constructions makes *STOS* a joy to work with.

STOS has more than 20 sprite-related commands alone, including collision detection, tests for location, pixel detection and synchronization with a scrolling background.

Other *STOS* commands include joystick read commands; numerous sound and music commands; screen fade, shrink, grow and appear commands; screen copy, pack, shrink and zoom, window and menu commands and many others; not to mention the usual BASIC command and function list—over 300 commands altogether.

To facilitate game construction, *STOS* also comes with—among other accessories—a sprite, font, icon and music designer (accessed from within the BASIC program itself!). There is also a "map" editor for creating games with scrolling backgrounds such as *Time Bandit* and *Gauntlet*. These accessories can be loaded as background tools or as BASIC programs to edit and change or run. Some of them are a tad awkward, but all perform reasonably well.

STOS has some unusual but powerful features. For example, there are 15 memory banks reserved for program use. These can contain sprites, screens, music, data, machine language programs, icons, character sets or be used as temporary workspaces. Banks are saved with your program—that's why the accessories can take music, sprites and so on from other *STOS* programs so easily. *STOS* also uses two levels of screens, a background screen for sprites and another for foreground display. "Physic" and "Logic" differentiate between two foreground screens, which can be exchanged with the SWAP command. Banks can also be reserved for other screens and exchanged with the display through a few simple commands.

There are commands to list and print bank information. *STOS* can switch back and forth between low and medium resolution within a program, without crashing. You can store up to four programs in memory at once and switch between them easily.

STOS has a "smart" file save feature: It automatically saves a file in the correct format according to the extension: .ASC for ASCII, .BAS for tokenized BASIC, .PRG to create a run-time program, .ACB for accessories, .MBK and .MBS for memory banks, .VAR for a list of variables and .PI1, .PI2, .PI3 or .NEO for *DEGAS* or *Neochrome* graphics.

Probably the only fault I can find with *STOS* is in the manual. It's not badly written, but the organization and layout are confusing and awkward. Commands are grouped by function (e.g., sprites), but are not in any order within a group (alphabetical order seems not to have occurred to the writers). This makes it difficult to find things by browsing. The "Guided Tour" section shows off some dazzling effects with simple commands, but doesn't tell how or why—a source of great frustration for the newcomer. You're not always told everything you need. For example: How many windows can be opened at once?

There are few graphics and screen shots, when many would ease comprehension problems. And there are a few typos that may cause a few headaches. For example, if you use their example *DIR$=[\ STOS"*, you'll get an error—the syntax is *DIR$ = [STOS"* to go down one directory level. Another: On page 164, it says 8 x 16 fonts can be used "happily" in all three resolutions, but on page 170, it says they're only usable in high and medium resolution.

Because of the clumsy manual, you'll find yourself jumping all over the place, trying to understand features, trying to find associated commands, but one saving grace is the abundance of example code. I only wish they'd included some examples of the results of that code.

STOS isn't a GEM program, so it doesn't use or have access to any GEM (or GEMDOS), AES or VDI calls. Nor can you get at the BIOS or XBIOS calls. It uses its own equivalents, although some, like the file selector, are rather inelegant. The menu commands do not equal those found in GFA BASIC; however, you can use icons, not just text in *STOS* menus. You can gain an extra 32K or so of workspace by auto-booting *STOS* rather than loading it through the GEM desktop.

Finally, you can easily create "run time" versions of your games for commercial release. There are special utilities and commands to do this from within the BASIC interpreter itself. These save a .PRG file and a reduced version of *STOS*, similar to GFA's run-only program.

STOS comes with three games: a horizontally scrolling train-race game, a *Galaxians* clone and a *Blockade* clone. The latter, called *Orbit*, is itself worth the price of the package. Each also has a construction set for designing your own screens and levels! All three demonstrate *STOS*'s amazing speed and flexibility plus they can be listed and changed. I added a high-score file-save to *Orbit* after a few minutes of listing and examining code for a suitable location.

Overall, for BASIC buffs and would-be game designers, *STOS* is an excellent buy and a fun alternative to the other BASICs. It's one of the better buys on the ST right now.

Parting shots

First the good news: I was amazed to discover that Borland—a major software publisher in the PC/MS-DOS world, known for its superb products such as *Sidekick*, *Paradox*, *Sprint* and the Turbo languages—has published Turbo C for the ST. Yes, an amazingly fast, powerful, beautifully crafted C! I watched a demonstration of it recently (*Calamus* was written in Turbo C). It simply outdoes everything else in the category.

Now the bad news: It's only available in Germany. Why? I don't know. If you want to find out, write to Borland, 4585 Scotts Valley Drive, Scotts Valley, California 95066. Apparently the company doesn't think the North American market is the right place to sell its own product. ■

Ian Chadwick is a Toronto-based freelance writer and editor, specializing in technical writing, software manuals and computer documentation.

FROM OVER THE BIG WATER

By Marshal M. Rosenthal

Computers don't live or die because of how they are made. Even if they have mice, run clock cycles around the competition and cost less, they can still disappear. What determines the longevity of a silicon machine is software availability.

In the beginning, the Atari ST enjoyed a lead over the "other" 68000 machine (not the Mac) for this very reason. It's a strong selling point to be able to say, "Hey, look at all the stuff you can get," and one that makes sense. But software houses are fickle; they look at cold, hard numbers and translate that into cold, hard cash. So it's no wonder that software here in the States became more of a trickle than a flood when ST sales slackened.

Enter Europe. Exit the philosophies of the U.S. market, such as "games are no good" and "they make a computer look silly." Like it or not, it's the games that sell—everybody likes to play *something* at some time on a computer. So while American companies pulled back on the software market, the European companies damned the torpedoes and steamed full ahead.

To understand the European software market is to understand some simple points:

1. The Europeans produce a lot of games—many more games than productivity software.
2. Europe includes many countries, all interacting. People are more aware of what's going on in their neighboring countries and thus are well informed.
3. Because the European countries are so closely tied, European software publishers target their products at both the local population and the populations of their neighboring countries.
4. Because of the competition formed by the "international" marketing, the price of software must be kept down.

The above points have insured a steady flood of software for the ST in Europe. And now more and more of the European software packages are becoming available in the U.S., with many U.S. software houses importing and selling them (often with small cosmetic changes for cultural differences—and too often with a higher price).

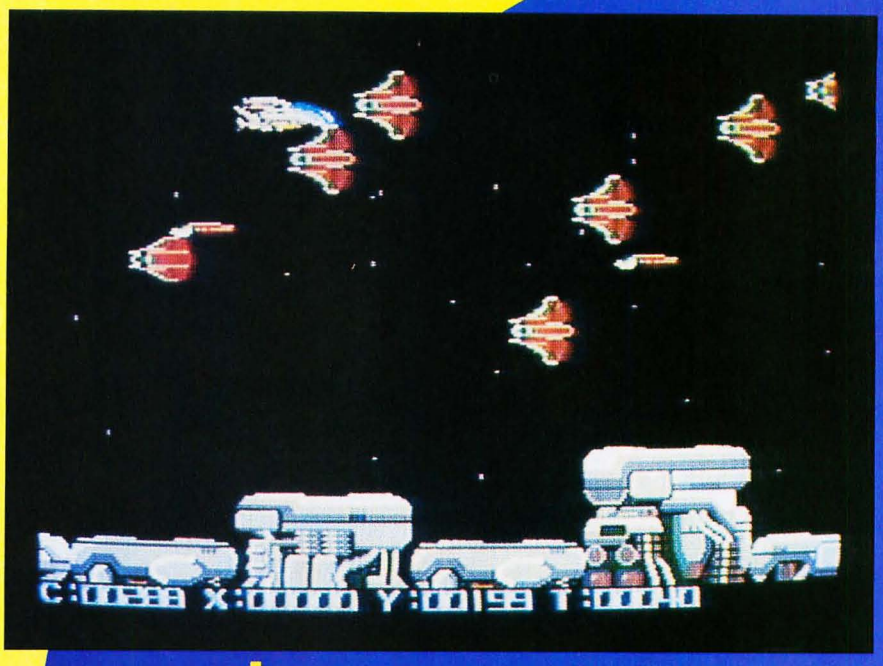
So let's start our exploration of the European software market with a look at some of the many ST games that are appearing from our cousins across the sea.

The fun stuff

Domark has already put out two games based on James Bond films. It's no surprise, then, to find them releasing *Live and Let Die*, an action game where you take your Q-equipped speedboat across waterways in search for the mysterious Dr. Big. Varied landscapes pass by as you progress, with the CIA dropping supplies to you by helicopter. Lots of mines, baddies and obstacles get in the way—so blast 'em all to bits. After all, you're the hero, right?

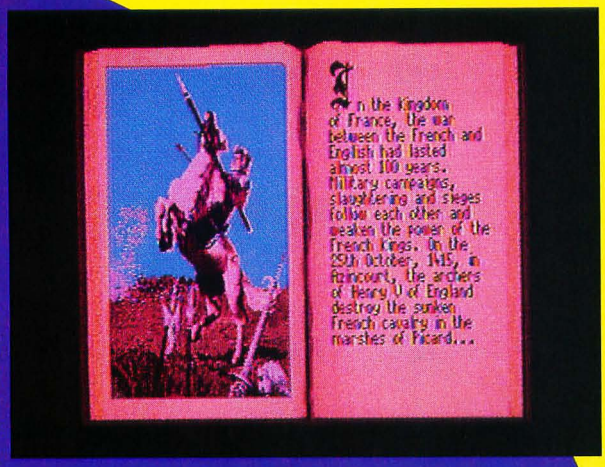
Digital Magic's *Trained Assassin* pits you against hordes of ghastly creatures as you blast your way through five deadly zones. Slick animation at 50 frames a second combine with scrolling action.

Or journey through magical lands in

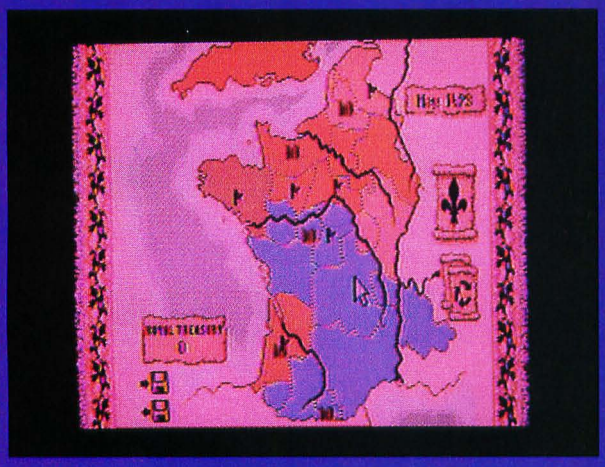


R-TYPE · ACTIVISION U.K.

LIVE AND LET DIE · DOMARK



JOAN OF ARC · SOFTGOLD



SPACE BALL · RAINBOW ARTS SOFTWARE

Scorpion, where the battle is on to rescue the princess. Kill alien creatures, recover magical weapons and gain numerous abilities in this action adventure. Lots of *big* animated enemies too.

Slip Stream from Microdeal features fast-scrolling fun, as you try to destroy alien ships in order to reach a power crystal lying in wait at the end of each stream. Three-dimensional shapes block the way, and you'll have to blast through them. Stereo sound, multiple options (like weather conditions) and random mission orders keep the game fresh and exciting.

In *Purple Saturn Day* from Infogrames, you train or go for full competition against 16 opponents in four games: *Ring Pursuit* has you flying around Saturn's rings, while *Tronic Slider* requires grabbing up energy globes. *Brain Bowler* and *Time Jump* will also tax your nerves and skill. Great animation and sound makes this a sure winner.

Those with less stamina might prefer *Action Service*. All you have to do here is complete boot-camp-like training. There are obstacle courses with barbed wire, as well as snipers to avoid and combat foes to take on with rubber bullets. Three routes keep you from memorizing what is coming next, and changes can be made as far as what shows up and when. Finally, sit back when it's all done and watch how well you did with the VCR feature.

Psyclapse (a division of Psygnosis) gives your reflexes a go with *Baal*. Fight against the demon and his evil underworld empire. Baal has ugly plans for the planet—so strap on the rocket pack, grab a laser gun and start searching for ways to mash him and his nasties into clam chowder. Easier said than done, of course, but a lot of fun is in store as you search for the dastardly machine that could spell doom to Earth. It may be a bit tough to find though; there are over 250 screens!

Activision U.K. released *R-Type* on a startled universe. The Bydo empire is out to take over *everything*, and of course, you don't like the idea. So off you go in your R-9 craft, which can acquire additional weapons by shooting alien ships. Horizontal scrolling keeps the action moving at a furious pace, helped along by teams of uglies. Each level ends (yeah?) when you wipe out a huge and hideous creature drooling at the prospect of having you for lunch.

Chyrsalis is a new software house, and its first presentation is *Prison*. Imagine being stuck in a dank and dirty penal colony on a forsaken planet. Of course, there's a way out—all you have to do is search through screen after screen until you gather the

clues that point to the location of the hidden spaceship. Looks like another blue Monday to me.

No one ever gets tired of the Man of Steel, and the forever 30-year-or-so-old dude has his hands full in *Superman—Man of Steel* from Tynesoft Computer Software. Terrorists are causing mayhem: earthquakes shake, rattle and roll, and volcanoes are belching. Yep—just another typical day. Lex Luthor and Darkseid are behind it all. *Help*! As our boy battles above, on and below the earth, it's a good thing he's got heat vision, super strength and lots of smarts. This ain't no picnic.

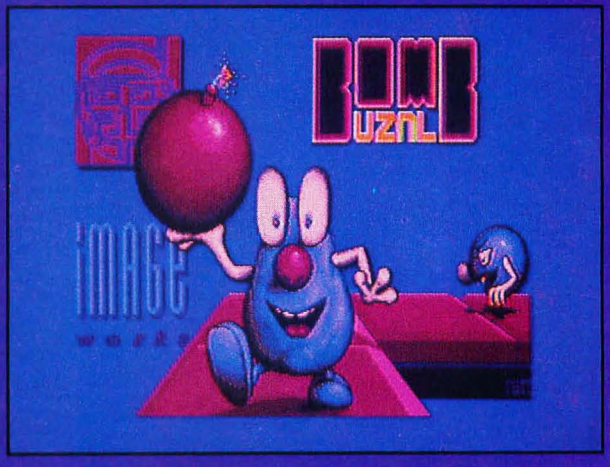
We all like cute, don't we? If you answered yes, *Bombunzal* is for you. Yes, it's another *Pac-Man* spinoff, but there's fun to be had as you direct this chubby little fellow along a platform of tiles to explode all the bombs without blocking off his route. Animation of the character is part of the fun here—watch him react to explosions. Meanwhile, friendly and unfriendly droids keep popping up to complicate the matter.

Rainbow Art's *Spaceball* is engaging and perhaps a bit reminiscent of that great-granddaddy of all, *Pong*. You control a bumper jet that has upward and downward movements. Besides fighting to touch the floating Para-Space Floaters, you must also gather the bonus symbols that appear—and don't let your opponent destroy the Astro bar behind your jet, or he'll gain points. It all makes for a simple-to-understand game that has addictive qualities. (Hey, does anyone remember *Warlords* for the 2600?)

Joan of Arc from Rainbow Arts has a little bit of *Defender of the Crown* mixed in with some solid strategy game play. The program first takes you through a series of digitized images that explain just how Joan got her start and why the French king decided to let her command his army in an effort to kick the English back to their own borders. Now it's time for Joan to do her thing while you handle the heavy thinking.

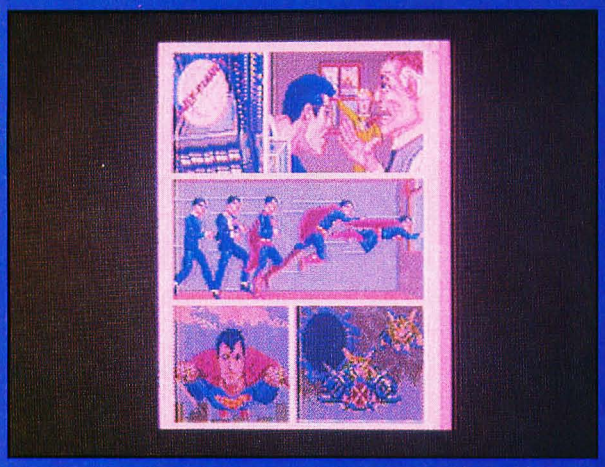
You are Charles, Dauphin of France. Joan must stop the English from taking Orleans and remaining in any of the provinces—or you can kiss off the crown. Warfare is only one of the tools to use: There's also diplomacy, espionage and "removal" services (having someone poisoned or kidnapped). Of course there are other alternatives, as well, such as enforcing royal justice or slapping on a tax or two. Each turn allows the dreaded English to become more entrenched, and you can view them coming over the walls of any castle that has fallen. But it's not all strategy. There are also action sequences, such as

RAINBOW ARTS SOFTWARE · **SPACE BALL**



IMAGEWORKS · **BOMBUNZAL**

SUPERMAN · TYNESOFT COMPUTER SOFTWARE



TALESPIN · MICRODEAL

fighting your way past archers into a town and engaging your troops with the English.

Getting a little more serious

It's not *all* games in Europe though. There's also a wide range of utility and productivity programs, even some hardware devices. Here's a few worth mentioning.

Talespin from Microdeal lets you create your own adventure games, and even though the end result is professional and polished, getting there doesn't require a degree in computer science. You will, however, need the ability to draw, so expect to spend some time with *DEGAS* or *Neochrome* before you start really using this program. Then import the images (up to 50 screens per game) into *Talespin* and clean them up with the included graphics options. Next you create a page for the image, then construct your plot for it. In the final game, a mouse click on the image will bring up a word balloon and game choices to follow.

Sound can also be incorporated into your game—even sampled effects. (Microdeal also makes a hardware device called Replay-4). The finished game is then saved as a stand-alone disk that boots up automatically.

Sprite Master from Soft Bits gives you the power to create your own sprites in low-resolution and even animate them. All controls are mouse-driven, logical and easily learned. The animation mode lets you move your *DEGAS*, *Neochrome* or *Paintworks* screens, with full control over direction and speed. The animation can then be saved for use with BASIC, C or assembly language programs.

Logitech's *Flair Paint* adds a positive note to ST drawing packages. Features include blitter support, icons that move along with the on-screen image, IMG format support, a selection of brushes, 16 colors and 36 patterns that can also be used as brushes.

Want more? How about Bezier curves that look *good*, rotation and re-sizing choices and key commands for menu selections. In addition, *Flair Paint* can be installed as an accessory and pulled down from within other programs.

Science fiction has always preceded reality, with computers being no exception. Before the advent of "talking" computers, the touch screen was frequently referred to, where the computer responded to your hand passing over it. In the real world, touch screens haven't fared quite so well, and the graphical interface and the mouse have taken over. But should you be interested, Eagle Computers lets you experience a real "hands on" approach to the ST.

Eagle's Touch Window consists of two transparent mylar sheets that are attached

to the front of the monitor. A touch closes an electrical contact that sends the coordinate information through the serial port at 4800 baud. Response isn't fantastic, though, because the system cannot quite keep up with you. Of course, a special program, called a driver, must first be run to inform the ST of what is going on. Then you can simulate the mouse with your finger, pen or whatever. There are also places marked at the bottom of the Touch Window that simulate mouse buttons.

The system will work with most programs, but the price might be a bit tough to swallow—about $750.

Desktop publishing is gaining popularity by leaps and bounds. One of the problems that faces someone using one of these programs is getting graphics from a page converted into digital form. You can go the digitizing route (if you have a video camera and the hardware), buy premade clip art or consider the Handy Scanner from Cameron.

Models 2 and 3 of the Handy Scanner give you good alternatives for black and white scanners. Both models resemble a slightly larger and elongated mouse, attached by cable to an interface box that plugs into the cartridge port (with an external power supply). Both have resolutions of 200 dots per inch, with Model 3 able to handle color and produce an image with up to 16 shades of gray. You just drag the scanner over the art—after installing the software program, of course—and the unit's sensors do the rest. Swaths about two inches wide mean that you may have to resort to multiple passes combined with cut and paste operations—but it sure beats drawing!

Till next time

That concludes this issue's romp through Europe. Keep in mind that many of these products—especially the games—may be available at your local Atari dealer. Give them a call; you may be pleasantly surprised.

Products mentioned in this article:

·**Baal**
Psyclapse sygnosis
Freepost, Liverpool
England L3 3AB
About $38

·**Bombunzal**
Imageworks
Headway House
66-73 Shoe Lane
London, England EC4P 4AB
About $38

Flair Paint
Logitech
166-170 Wilderspool Causeway
Warrington, England WA4 6QA
About $64

·**Handy Scanner**
Cameron UK Ltd.
108 New Bond Street
London, England W1Y 9AA
Model 2—About $425
Model 3—About $635

·**Joan of Arc**
Softgold
Unit 2/3 Holford Way
Holford, Birmingham
England B6 7AX
About $38

·**Live and Let Die**
Domark
204 Worple Road
London, England SW20 8PN
About $38

·**Prison**
Chrysalis
England
About $38

·**Action Service**
Infogrames
Mitre House
Abbey Road, Enfield
Middlesex, England EN1 2RQ
About $38

·**Purple Saturn Day**
Infogrames
Mitre House
Abbey Road, Enfield
Middlesex, England EN1 2RQ
About $45

·**R-Type**
Activision U.K.
England
About $38

·**Scorpion**
Digital Magic
103 Mersey Road
West Bank, Widnes
Chesire, England WA8 ODT
About $45

·**Spaceball**
Rainbow Arts Software
GmbH Hansaallee 201
4000 Dusseldorf 11
West Germany
About $38

·**Sprite Master**
Soft Bits
5 Langley Street
London, England WC2H 9JA
About $45

·**Superman—Man of Steel**
Tynesoft Computer Software
Addison Industrial Estate
Blaydon, Tyne & Wear
England NE21 4TE
About $45

·**Slip Stream**
Microdeal, Ltd.
Box 68
St. Austell, Cornwall
England PL25 4YB
About $38

·**Talespin**
Microdeal, Ltd.
Box 68
St. Austell, Cornwall
England PL25 4YB
About $90

·**Touch Window**
Eagle Computers
Glamorgan House, 2nd Floor David Street
Cardiff, England CF1 2EH
About $750

·**Trained Assassin**
Digital Magic
103 Mersey Road
West Bank, Widnes
Chesire, England WA8 ODT
About $45

Marshal M. Rosenthal is a New York-based writer and photographer whose work takes him throughout the world. His written/photographic projects have appeared in England, Germany, Spain, Mexico, France and the U.S. He is presently assembling a gallery exhibition on "Children of the Philippine Islands," the result of a four-week photo-shoot sponsored by Kodak.

STEP 1: Understanding Atari GDOS Part II

by Maurice Molyneaux

I hope the past 30 days has been enough time for you to digest all the information about GDOS provided in last issue's *Step 1*. This time we'll get away from what GDOS is and discuss using it, and problems associated with it. In any case, you may want to keep last month's issue handy in case you find a need to refer back to part one of this subject.

Giving it the boot

As stated last issue, GDOS is a program that must be installed by placing a copy of it in the AUTO folder of your boot disk and starting or resetting your machine. When the system is booted or reset, it runs all programs from the AUTO folder before going to the GEM Desktop. Remember, it will only run .PRG programs. If a program has a different extender, it won't run (so you can't use the .APP [Application] extender). You should know what your boot drive is, but just in case, we'll cover that now (if you know this, and most of you probably do, skip over the next paragraph).

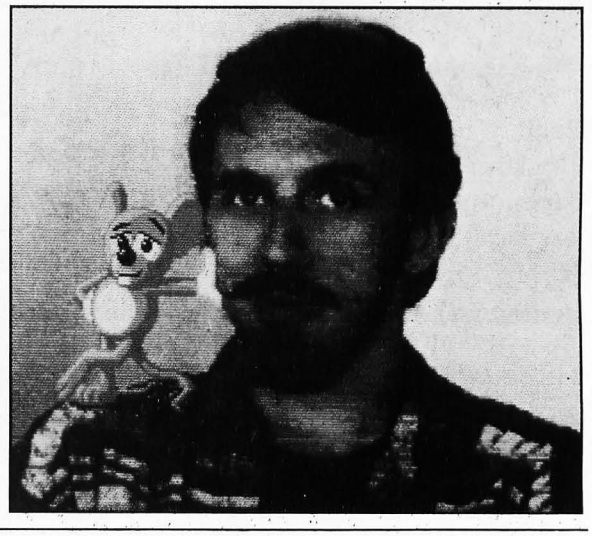
The boot drive is where your ST looks for the AUTO folder, desk accessories and a DESKTOP.INF file. If your system is floppy-based, then your boot disk is drive A (the internal drive on a 520STfm, 1040ST or Mega). If you have a hard disk, it is possible that it is the boot disk, but whether or not this is the case depends on how the hard drive is configured. Every hard drive I've ever used for the ST can be set up to auto-boot. If auto-booting is enabled, the first partition of the hard disk, usually drive C, is the boot drive. If not, it will be drive A, as usual. In the case of some hard-disk driver software, the hard disk can be treated as the boot drive, but this will only happen if a special hard-disk driver program is present on the A drive, which forces the system to turn around and boot from the hard disk.

If you have an auto-booting hard disk, but are running your GDOS software from floppy, you can force the system to boot from the floppy (and thus run GDOS from the AUTO folder there) by simultaneously pressing and holding down the alternate, shift and control keys while booting the machine (if your machine has the Mega ROMs you should not press these keys down until the floppy drive light comes on, otherwise the system won't boot from the floppy).

Be warned, however, that forcing this type of boot can, with some hard-disk driver software, cause the system to forget the hard disk is present. You will have to test this on your system to see if this is the case. If it turns out that booting in this fashion does cause you to lose access to the hard disk, you will either have to place a copy of your hard driver's booting program to the AUTO folder on the floppy disk of the GDOS application, or install GDOS on your hard disk (which is the easier way to go).

If you're using a floppy-based system, you can probably boot with a back-up copy of the GDOS application you are planning to run, as chances are it will already have an AUTO folder and GDOS set to go automatically. This might be fine most of the time, but if you want to change any aspect of what GDOS does and uses (such as adding or deleting fonts and device drivers), or if you want to have more than GDOS alone run from the AUTO folder, you'll probably want to put it on another disk, along with other AUTO folder programs, desk accessories, etc. This is also the reason hard-disk users will probably want to put GDOS on their hard drives.

To install GDOS on a disk other than the application disk the first thing you need to do is copy it to the AUTO folder of your boot disk (if you don't have an AUTO folder on your boot drive, create one), and that does it for the program. Next, you should make sure that GDOS's ASSIGN.SYS file is in the root directory of the



boot drive, and that the drivers and fonts (discussed last time) are in the directory specified by the ASSIGN.SYS file. If, for any reason these files are not where the ASSIGN.SYS file says they are, you will either have to move them to the correct place or edit the ASSIGN.SYS file to reflect their actual location. (For information on the structure and editing of an ASSIGN.SYS file, see last month's article).

Once all this is done, you should be ready to roll. You just boot your system and GDOS should install (you'll see a message stating as much during the boot-up process). You can then run your GDOS application and everything should be hunky-dory.

"Should be" are the operative words. If your GDOS application uses GDOS fonts, try to see if they are available. If none are, either none were present or the ASSIGN.SYS file told GDOS to load them from the wrong place (or contained the wrong filenames). If the application stores its data as a Metafile (see last issue) it probably uses a META.SYS driver to do so. If you find you cannot save your work, this may be the problem. For example, if *Easy Draw* cannot find the META.SYS file, when you attempt to save your work, an alert box will appear stating as much, and giving you the option of retrying or cancelling. If META.SYS is loaded from a floppy, it's possible that you have the wrong disk in the drive. If so, insert the correct disk and try again (if the program you're using permits). If these files were to be loaded off of your hard disk, then the problem is probably with the ASSIGN.SYS file and the path it sets for GDOS to load such files from.

Every silver lining

. . . is accompanied by a dark cloud. In this case, the silver lining is the power that GDOS gives (multiple font styles and sizes, resolution-independent images, Metafiles that can be ported from one GDOS application to another). The dark cloud is the many problems with GDOS and its use.

The first and most noticeable problem many GDOS users encounter is that of speed. Because GDOS is a "patch" to the ST's OS, it is pretty much invisible to the user. But just because you can't see it doesn't mean it isn't there and working. The trouble with GDOS is that it often slows the entire system down. Windows don't open quite as fast (particularly in a GDOS application), cursors don't move smoothly. The whole system seems to bog down and act sluggish. It can be a bit of a problem to move—and particularly fine-position—elements when the system acts so clunkily. You move the mouse a bit, but the cursor doesn't budge, so you continue to move it, and *blip!* The cursor suddenly jumps over—too far.

This problem was the subject of intense discussion, and, on the DELPHI telecommunication system, the target of a heated debate. The word from Atari was first that the GDOS slowdown was a myth, and later that it happened only on some machines. It later became evident that the problem was not with a handful of machines, but with GDOS itself. For a time the slowdown problem was such a common point of discussion that Andy Eddy (associate editor for ST-LOG) bestowed upon GDOS the nickname "G-DOZE," which stuck for quite a while.

Beyond the slowdown problem there are others. GDOS does not seem to like some programs. We're not talking GDOS applications here, but non-GDOS programs. In some cases, if such a program is run while GDOS is in memory, either it will be prone to failures or sometimes won't function properly at all. For example, DC-FORMAT will crash if GDOS is present. Usually, this is the fault of the program being run.

But these problems are dwarfed by the biggest annoyance that GDOS presents; namely, that it can load one and only one ASSIGN.SYS file. This may not seem to be much of a problem, but if you routinely use several GDOS applications, it can become a real migraine. The ASSIGN.SYS file used by *Easy Draw* is not the same as the one used for *DEGAS Elite* or for *Timeworks Desktop Publisher.* Sure, it's possible to create one all-purpose ASSIGN.SYS file containing all the fonts and device drives used by all your GDOS applications, but that file would be large, unwieldy, and not necessarily work as you intended it to.

The real pain here comes because GDOS reads the ASSIGN.SYS file only once, at boot-up. It cannot and does not read it again during a work session. Therefore, if you had multiple (different, and with different names) .SYS files, it would only read one (the one named ASSIGN.SYS), and ignore the rest. If that file was for *Easy Draw,* it would not help you if you were working with *DEGAS Elite.* The only way to get GDOS to load a different .SYS file is boot with a disk containing GDOS and a different .SYS file. On a hard drive you would be forced to rename the one you want to use to ASSIGN.SYS (after changing the current one to some other name) and then reboot! Yup. If you want a different ASSIGN.SYS file, you have to reset or restart your ST. This is not a very clever or convenient system. Having to rename .SYS files makes it bothersome.

Some solutions

How can you avoid these problems? Well, you could always just throw GDOS out the window and forget it. But no, that's a bit like curing the bug by erasing the program. There *is* a solution to just about every problem listed, but it means using a commercial program that serves as a replacement for GDOS. More on that later, right now let's talk about what you can do to alleviate problems for minimal cost.

The GDOS slowdown is caused, primarily, because GDOS runs in a constant loop, and monitors the ST's TRAP #2 vector. I'd better explain this: "TRAP" is the 68000 assembly language term for an interrupt/exception. The ST has 16 TRAPs; #1 is used by and for GEMDOS, #2 the VDI and AES (Virtual Device Interface and Application Environment Services, respectively, discussed in *Step 1,* January 1989 ST-LOG), and so on.

GDOS monitors TRAP #2, looking for certain VDI calls. The trouble is that the VDI is always active, reporting keyboard activity and mouse movement, amongst other things, so GDOS is watching all of this even though most of it is unimportant to GDOS itself. This slowdown is not all that noticeable if GDOS and a GDOS application are the only things in memory, but if you add any other programs that make use of the TRAP #2 call, that bogs the system down because of the amount of "traffic" going through that interrupt vector. The easiest way to see this slowdown is to load a few desk accessories. These make a lot of AES calls, which use TRAP #2, and thus add to what I call the "TRAP #2 bottleneck."

Why this happens is not just due to the business of TRAP #2, but can be attributed to the way GDOS monitors the TRAP. The code for this is not optimized and takes more time than it should. This is partly because GDOS was written in C and not in assembly language, which would have been faster.

The way around this is not to use accessories unless you really have to. This will ease the amount of VDI and AES calls and allow GDOS to run faster. I don't consider this to be a great solution, but a workable one.

Another solution is to boot with GDOS when and only when you intend to use a GDOS application. If you don't want to use GDOS, you can either boot with a disk that doesn't have it or you can use one of those AUTO-folder-program-selection utilities (which let you choose which programs in an AUTO folder to run), like *Desk Manager.*

The next problem is that of multiple ASSIGN.SYS files. If you're working with a floppy-based system, the easiest way around this is to keep your GDOS applications on different disks, and on each of those disks place an AUTO folder containing GDOS, and place the ASSIGN.SYS file for that program in the root/main directory of that disk (also be sure to copy the fonts and drivers to that disk as well, and make certain that the ASSIGN.SYS file sets the proper path for them). When you want to use a given GDOS application, you just boot with the appropriate disk, and *voila!*

If you have a regular boot disk or boot from a hard drive, there is another solution. Both the shareware and commercial versions of *Desk Manager*, by Charles F. Johnson (not the version originally published in ST-LOG #16), have a feature which allows the user to choose an ASSIGN.SYS file at boot-up. The *Desk Manager* program runs from the AUTO folder and allows the user to select which AUTO folder programs to run (as mentioned above), what accessories to load, and even choose between multiple ASSIGN.SYS files.

This is done by copying all your different ASSIGN.SYS files to a subfolder in the AUTO folder called DESKMGR and giving them descriptive names with a .SYS extender, like EASYDRAW.SYS or DEGAS.SYS. If a program starting with the letters GDOS is active in the AUTO folder, you will be allowed to select any one of the .SYS files you copied to the *Desk Manager* folder. When you have made your selection, *Desk Manager* copies the selected file to the boot disk's root directory and names it ASSIGN.SYS. Then, when GDOS runs, it will find and use the file you selected. In this manner, you can keep several ASSIGN.SYS files on a single disk and never have to manually rename them again.

Shareware versions of *Desk Manager* are available on all the major on-line services and from some user group libraries. The latest, commercial version is *not* freely distributable and is one of the many programs on the *CodeHead Utilities* disk from CodeHead Software.

These two solutions ease the slow down and eliminate the annoying renaming problem, but it does not eliminate the forced rebooting in order to switch ASSIGN.SYS files, nor does it eliminate the malfunctions and crashes some software experiences when GDOS is in RAM.

The best bet yet

If you use a lot of GDOS applications, the biggest favor you can do for yourself and your ST is to shove GDOS into the archives (maybe even flush it down the toilet) and forget it. Of course, you have to replace it with something, and the answer is to buy *G+Plus* (the "+" is silent; consider it a stylized hyphen), from CodeHead Software. (I don't usually like to plug commercial software, but there is simply nothing else out there that can do what this program does.)

G+Plus rectifies just about every one of GDOS's failings while maintaining a compatibility level that is nothing short of miraculous. The slowdown is gone with a vengeance, ASSIGN.SYS files are no longer a hassle, and most of the problems with other programs have been eliminated.

This program will cost you $34.95 (list), but if you use a lot of GDOS applications it's worth every penny of that price.

GDOS errata

It has been said that there has been more myth, mythconception, mythunderstanding and downright mythinformation (sorry) about GDOS than about any other part of the ST or any of its software. A lot of the blame for this falls upon the shoulders of Atari, which took a great deal of time to produce a fairly flawed GDOS, and which hasn't provided developers with the kind of detailed information and guidelines they need.

Then again, many developers have failed to adequately cover the use of GDOS in their manuals, even though GDOS comes with their programs and did

not originally come with the ST. In many cases where GDOS *is* discussed in a product manual, the information presented is inaccurate. I have seen manuals that explain how to set up an ASSIGN.SYS file for a program, and the instructions are wrong!

Here are a few common myths and misconceptions we can shoot down:

Myth: *A GDOS application must have GDOS present to run.*

Sometimes, but not always. The most common instance of users assuming this is with *DEGAS Elite*. That program uses GDOS for one thing and one thing only: to load fonts other than the system font. So, if you don't need the extra fonts, you don't need GDOS. Other programs *do* need GDOS to function. As I mentioned earlier, *Easy Draw* cannot save a .GEM file if it cannot find the META.SYS driver loaded by GDOS. A little experimenting will show you which GDOS programs you must use GDOS with and for which functions, and which ones you can safely use without GDOS.

Myth: *GDOS is a memory hog.* GDOS is a fairly puny program, and it does not load fonts and drivers until they are called for. The fact that it must load multiple font files for multiple sizes of a single font style is the one thing that can make it hog memory. And, as stated last issue, GDOS *can* and *does* unload fonts, contrary to the prevailing notion which says it can't or doesn't.

Final recommendations

If you use GDOS, let's sum up with the following pointers:

—Don't install GDOS if you're not going to use a GDOS application. Using a program that allows you to select which AUTO programs to run makes this easy.

—If you are using GDOS, limiting or eliminating desk accessories can help alleviate the "G-DOZE" slowdown syndrome.

—Don't make an all-purpose ASSIGN.SYS file for multiple applications. Get a program like *Desk Manager* so that you can pick and choose .SYS files.

—If you use a lot of GDOS applications, consider purchasing G+Plus, because it will make your life a lot easier.

And that's all. I hope this article and its predecessor have been useful to those of you who use (and probably have suffered with) GDOS. As always, I've tried to cover the topic as exhaustively as possible and present all the information I think end users can use. But it's possible I might have missed something or even made a mistake. So if you have any comments or corrections, please let me know about them, in writing, care of this magazine. Any omissions or errors reported will be mentioned in future columns.

Addenda: In response to the "Of Mice and Megabytes" articles in the November and December '88 issues, I have received a few pieces of written mail and quite a few electronic mail messages from readers who were interested in obtaining one or both of the animated videotapes mentioned in those articles. To everyone interested, I'm sorry, but the answer is no. This is because I am still trying to sell the *Balance of Terror* game demo, so I cannot release it. And, in regard to the *Art & Film Director* demo, that tape is distributed by Epyx Inc., which uses it for dealers, distributors and trade shows. If at any time either of these become available, I'll let you know here. ▪

Maurice Molyneaux designs game graphics, consults for software companies and creates animated cartoon productions using his ST. Despite a ridiculously French name, he was born in Vicenza, Italy, and denies vicious rumors that he eats escargots and calamari while computing. His DELPHI username is MAURICEM.


B$=A$:R=R+1:PUT 1,R:GOTO 110
PRINT:PRINT "ALL DONE!"
data 00,00,00,32,00,0E,00,00,00,0
1250 data 00,12,00,19,00,00,00,00,00,0
1260 data 00,50,00,13,00,10,00,08,00,0
1270 data 00,FF,11,00,00,02,00,00,00,1

ASSEMBLY LINE: FILE HANDLING, Part II

by Charles F. Johnson

With this edition of *Assembly Line,* I'll be taking over the writing chores from Douglas Weir (who's done a terrific job up to now). Douglas's last column dealt with opening disk files and reading data from them. This issue's column will take up more or less where he left off and explain how to access GEM's File Selector through assembly language.

This month's example program actually does something useful: It copies a file from any directory to any directory. To do this, we need to open files, create new files, read and write data and close everything properly. To make things a little more exciting, the example program will use the GEM File Selector to choose the source and destination filenames.

Danger, Will Robinson!

I would be remiss in my duties as a columnist if I didn't take a few moments here for the obligatory warnings. Be *very* careful when experimenting with untested code that writes to disk. As Douglas mentioned in his last column, you should do your testing on a "work" disk (one that you don't care about), and definitely not on a hard disk or a floppy that contains any data you don't want to lose. It's very tempting to just type in that code and run it. I know, but trust me, the very first time you let your guard down, Murphy's Law will jump out of the bushes and grab you! It's only a few seconds' work to pop out the disk with your source code and pop in your work disk—a few seconds that can save you hours of future grief.

The Incredible Shrinking Program

When GEMDOS loads a program into memory and starts it, the program is allocated of all the available memory in the computer. In other words, when our program runs, it doesn't matter where we put our *.end* statement. As far as GEMDOS is concerned, we own all of the memory from the start of our program to the last byte of free RAM, and *no one else can use it.* Which means that if a desk accessory that's running concurrently with our program needs to grab some temporary memory to perform a function, it will not be able to do it.

Also, some operating system calls (particularly some of the AES/VDI calls) need to use some temporary memory and won't work if they can't get it. Therefore, the first thing any program should do when it runs is tell GEMDOS to take back all the rest of the available memory and reserve only the actual memory the program needs to run. The GEMDOS call that does this is called *Mshrink*, because it "shrinks" the memory used by the program and frees up the rest for use by any other process that needs it.

The *Mshrink* call passes two parameters on the stack, both longwords: the starting address of the area to reserve and the number of bytes to reserve. The starting address will usually be the base page of the program. You may remember from the last *Assembly Line* that when GEMDOS executes a program, the address of the program's base page can be found at *4(sp)*. The first thing our example program does is get this address from the stack and put it into register *d1*. Since the program's *.text*, *.data* and *.bss* segments immediately follow the base page, we can find the total number of bytes in our program by subtracting the address of the base page from the address of the end of the program (which is given the label *prgend*). These values are then passed to the *Mshrink* call.

The second line of code in our program is responsible for setting up the stack frame that our program will use. (Earlier *Assembly Line* columns have explained the usage and maintenance of the 68000's stack pointer.) When a program is run, GEMDOS sets aside a small stack area for the program's use; but this "default" stack is of very limited size and may not be adequate for many purposes. Therefore it's usually better to define an area in your *.bss* segment (the uninitialized data segment) that will be used for your program's stack. The new stack frame is set simply by loading its effective address into register *a7* (which most assemblers will also let you call *sp*).

Who's afraid of the big bad File Selector?

As mentioned above, this month's example program uses the GEM File Selector to select two filenames, the source and destination files for a copy operation. The File Selector is a handy "canned" routine in the ST. It lets you quickly find files in any drive or directory and select one of them for use by a program without ever touching the keyboard. It's a godsend for those who hate to type long pathnames when they need to access a file (which includes me). And fortunately for the assembly programmer, *fsel_input* is also the easiest-to-use function in the entire AES library. (I like to think of the File Selector as the programmer's best friend.) I'm going to save in-depth discussions of AES and VDI for future columns; to use the File Selector, knowing some of the bare essentials will do for now.

It doesn't take a lot of work to set up the necessary data structures for *fsel_input*. Near the end of our example program, in its *.bss* segment, you will find six arrays labeled *contrl*, *global*, *intin*, *intout*, *addrin* and *addrout*. These arrays are used by the system when you call any AES functions. Just before the *.bss* segment of the example program, you will also find the label *aespb*, that contains a table of pointers to the six AES arrays. This table of pointers is called the "AES parameter block" and is used to tell the system where to find your data arrays.

Each AES function has a specific number; to call a function, you place its number (or "opcode") in the first word of the *contrl* array. Depending on which AES function you're using, other parameters in these arrays may also need to be set before the call. After initializing the arrays, you actually execute the AES call by loading the address of the AES parameter block (*aespb*) into register *d1* and the number $C8 into register *d0*, and performing a *trap* #2 instruction. ($C8 is a "magic number" that tells the system that this is an AES call; VDI calls also go through *trap* #2, and the system differentiates between the two types of calls by checking the magic number in *d0*. The VDI's magic number is $73.)

The *fsel_input* call requires two parameters (both longwords) to be passed to it in the *addrin* array. These parameters are the addresses of the two text strings that will be displayed on the File Selector's "Directory:" and "Selection:" lines. The selection line should be a minimum of 13 bytes long, just long enough to hold a zero-terminated filename without any path. It doesn't need to be initialized. In our example program, the selection line is labeled *copy_file*.

The directory line should be long enough to contain nested subdirectories; 80 bytes is a reasonable amount to set aside. In our example program, the directory line is labeled *copy_dir*. Unlike the selection line, however, the directory line needs to be set up before passing it to *fsel_input*. After the *Mshrink* call, our program initializes the directory line.

Final approach to the File Selector

First, we use GEMDOS function $19 (*Dgetdrv*) to find out the current drive. This function returns a number from 0 to 15 in *d0*, signifying drives A through P. We convert this value to an ASCII letter by adding the ASCII value of the letter "A" (65) to it. Then we use the *lea* instruction to load the effective address of our directory line (which is an 80-byte area in our *.bss* segment labeled *copy_dir*) into register *a6* and move the ASCII drive byte from *d0* to the directory line, using indirect addressing with post increment. Then we move in the colon which should always follow the drive letter, again with post increment addressing. This leaves us with *a6* pointing to the first byte past the colon.

Now, we use GEMDOS function $47 (*Dgetpath*) to get the full pathname of the current directory. This function passes two parameters. The first parameter is a word signifying the drive whose path is being determined. To get the path for the current drive, this value should be 0; otherwise it should be a value from 1 to 16. (Notice that this value is one greater than the corresponding values returned from *Dgetdrv*.) The second parameter for *Dgetpath* is a longword address where it should store the returned pathname string. Since *a6* still contains the address immediately following the colon in *copy_dir*, we simply push *a6* on the stack and the system will put the current path into our directory line for us, terminated nicely with a 0.

We're still not finished setting up the directory line—now we have to add the rest of the search specification to the end of the string. Since we have no way of knowing how many characters are in the current path, we have to search forward

from the beginning of our directory line until we find the 0 at the end. This is done with a short, simple loop that tests each byte in the string (again, using indirect addressing with post increment) and repeats until the zero flag is set. When we find the end, we once again use post-increment indirect addressing to copy the rest of the search spec ("*.*") on to the end of the string, with another simple loop which repeats until the zero flag is set.

Behold the dreaded File Selector!

Once this directory line initialization stuff is out of the way, our path (pardon the pun) is clear to call the File Selector and let the user enter the source filename for our copy operation. To do this, we've created a subroutine called (imaginatively enough) *fsel_input*. This subroutine takes two longword parameters: the address of the directory line (*copy_dir*) and the address of the selection line (*copy_file*). The parameters are passed in registers *a4* and *a5*, respectively.

The first thing the *fsel_input* subroutine does is set up the AES *contrl* array. All AES calls require five words to be placed in the *contrl* array, as follows:

contrl = AES function number. (Opcode)

contrl + 2 = Number of words passed in intin.

contrl + 4 = Number of words returned in intout.

contrl + 6 = Number of longwords passed in addrin.

contrl + 8 = Number of longwords returned in addrout.

For the *fsel_input* call, these five words are:

```
contrl = 90
contrl + 2 = 0
contrl + 4 = 2
contrl + 6 = 2
contrl + 8 = 0
```

The *fsel_input* subroutine first places these values into the *contrl* array, and then moves *a4* and *a5* into *addrin* and *addrin* + 4. (Why "+4"? These are longwords, remember?) When the arrays are set up, we call the subroutine labeled *aes*. This routine moves the address of the AES parameter block (described above) into register *d1*, the AES magic word $C8 into *d1*, and executes a *trap* #2 instruction. At this point, if everything has been done right, the File Selector will appear in all its debatable glory, showing the files in the current directory of the current drive that match the search specification in our directory line. The *trap* #2 call will not return until the user of our program selects a file, or clicks on the File Selector's cancel button.

Please note that while the File Selector is active, it actually changes the data in our directory and selection lines! When we return from the *fsel_input* call, the file and path selected by the user will be right there, waiting for us to make a filename out of it. But first, we have to find out whether the user really selected a file, or whether she/he clicked on the cancel button.

The names of the AES data arrays reflect their functions; *intin* contains the *integer input* parameters, *intout* contains the *integer output* parameters, *addrin* contains the *address input* (longword) parameters and *addrout* contains the *address output* parameters. According to our table above, *fsel_input* returns two parameters in the *intout* array. The first element of *intout* will contain an error status code and will be 0 if an error occurred and greater than 0 if no error occurred. The array element *intout* + 2 will hold a value that indicates which button the user selected. If *intout* + 2 contains a 0, the cancel button was selected; if it contains a 1, the OK button was selected or the file chosen was double-clicked.

Our *fsel_input* subroutine, after returning from the *aes* routine, checks the contents of *intout* + 2. If any value other than 1 is present, the routine sets *d0* to −1 and returns, signalling to the caller that an exit condition exists. If this happens in our example, the code then prints a farewell message and exits. If *intout* + 2 contains a value of 1, the user clicked on the File Selector's OK button, or selected a file by double-clicking it, and *fsel_input* returns with *d0* set to 0.

Next we test the first byte of our selection line (using the *tst* instruction, which, as you recall sets the 68000 condition codes according to the contents of an effective address operand without changing it). If the first byte is 0, the user clicked on the OK button, but with a blank selection line; in this case, we again return with −1 in *d0*, causing the calling routine to exit, stage left. If the first byte is not 0, we have a valid filename on the selection line, and we'll return with *d0* set to 0. (Note that in our example program, we aren't handling any errors returned in *intout*.)

Constructing a filename

Now that we got out of *fsel_input* in one piece, what do we do? As mentioned above, the *fsel_input* call changes the data in the directory and selection lines. Our job now is to parse this data and build a full pathname out of it, which will be the source filename for our copy operation.

We're going to store the full pathname of the source file in a location in our *.bss* segment labeled *source*. Since we'll also need to construct a full pathname for the destination file, we've written a subroutine for this purpose called *make_name*. This routine takes three parameters, passed in registers *a0*, *a1* and *a2*: the address of the area in which to build the complete pathname (*source*), the address of the *fsel_input* directory line, and the address of the *fsel_input* selection line.

The *make_name* subroutine first copies the entire directory line to *source*, until it finds the zero terminator. Then it searches backward (using indirect addressing with pre-decrement) until it finds the last backslash in *source*, and moves the contents of the selection line to one byte past this location. When this is complete, the full pathname of the source file is contained in *source*.

The next section of code in the example program repeats the above steps to get the destination file from the user and build a complete pathname at the location labeled *dest*, by passing different parameters to the *fsel_input* and *make_name* subroutines.

To be continued

Hmmm. Looks like our discussion of *fsel_input* has gobbled up all the *Assembly Line* space for this issue. Next month, we'll continue taking apart our example program and maybe even get to discussing the file read/write routines. Until then, type in the example program, examine the code for yourself, and think about ways to improve it. ■

Charles F. Johnson, by using some as yet undiscovered laws of nature, has managed to find the time to be both a professional musician and a professional programmer. In his musical career, he has played with such artists as Chicago, George Duke, Al Jarreau and Stanley Clarke. His programming accomplishments include Mouse-Ka-Mania, Desk Manager, ARC Shell *and, along with his partner John Eidsvoog,* G + Plus *and* MultiDesk. *He and John are the owners of CodeHead Software.*

ASSEMBLY LINE
Listing 1:
Assembly

```
* COPY1.S
*
* Example program for Assembly Line
*
* Use the GEM File Selector to copy a file
*
* By Charles F. Johnson
* Copyright 1989 ST-Log
*

* Last revision: Friday, February 3, 1989  10:18:34 am

        .text

        move.l  4(sp),d1        ; Get address of basepage
        lea     ustack,sp       ; Set up my stack
        move.l  #prgend,d0      ; Address of end of this program
        sub.l   d1,d0           ; Get the length of this program

        move.l  d0,-(sp)        ; Push length of area to reserve
        move.l  d1,-(sp)        ; Push start address of area
        clr     -(sp)           ; Not used, but necessary
        move    #$4A,-(sp)      ; Mshrink
        trap    #1
        lea     12(sp),sp       ; Tidy up the stack
        tst     d0              ; Error?
        bpl.s   getdrv

        lea     cantshrnk(pc),a5
        bsr     print
        bra     byebye          ; If this fails, we flee in terror

* Set up default path specification for fsel_input

getdrv: move    #$19,-(sp)      ; Get current drive (returned in d0)
        trap    #1
        addq    #2,sp
        add.b   #'A',d0         ; Convert to ASCII
        lea     copy_dir,a6     ; Get address of fsel_input path string
        move.b  d0,(a6)+        ; Put in the drive letter
        move.b  #':',(a6)+      ; Followed by a colon

        clr     -(sp)           ; Get path for current drive
        move.l  a6,-(sp)        ; Put it right after the colon
        move    #$47,-(sp)      ; Dgetpath
        trap    #1
        addq    #8,sp

        lea     copy_dir,a0     ; Address of directory
.loop:  tst.b   (a0)+           ; Find the end of the pathname string
        bne     .loop           ; Loop til we get a zero
        subq    #1,a0           ; Back up one character
        lea     spec(pc),a1     ; Address of search spec string
.loop1: move.b  (a1)+,(a0)+     ; Tack the search spec to the end of the path
        bne     .loop1          ; Keep moving til we get a zero

        lea     title(pc),a5    ; Clear screen and print title
        bsr     print

* Get the first filename (the source file)

        lea     src_msg(pc),a5
        bsr     print

        lea     copy_dir,a4     ; Address of path line
        lea     copy_file,a5    ; Address of filename line
        bsr     fsel_input      ; Call fsel_input
        bpl.s   src_2           ; Returns with condition code set

fsel_cancel:
        lea     cancel(pc),a5
        bsr     print
        bra     byebye

src_2:  lea     source,a0       ; Location which will hold the source filename
        lea     copy_dir,a1     ; Get address of fsel directory line
        lea     copy_file,a2    ; Address of fsel selection line
        bsr     make_name       ; Construct the full pathname

* Now get the destination filename

        lea     dst_msg(pc),a5
        bsr     print

        lea     copy_dir,a4
```

```
        lea     copy_file,a5
        bsr     fsel_input
        bmi     fsel_cancel

        lea     dest,a0
        lea     copy_dir,a1
        lea     copy_file,a2
        bsr     make_name

        move    #0,d5           ; Search for normal files
        lea     source,a5       ; Address of source filename
        bsr     fsfirst         ; Search for it
        beq.s   get_free

        lea     cantfind(pc),a5
        bsr     print
        bra     byebye          ; If we didn't find it, bye bye

get_free:
        move.l  #-1,-(sp)       ; -1 = Return free memory
        move    #$48,-(sp)      ; Malloc
        trap    #1
        addq    #6,sp

        cmp.l   dta+26,d0       ; Compare size of file with amount of free RAM
        blo.s   no_memory

        move.l  dta+26,-(sp)    ; Allocate enough memory to hold the entire
        move    #$48,-(sp)      ; file
        trap    #1
        addq    #6,sp
        tst.l   d0              ; Test for error
        bne.s   gotram

no_memory:
        lea     no_ram(pc),a5   ; If zero, scram
        bsr     print
        bra     byebye

gotram: move.l  d0,copy_buffer  ; Save address of buffer

        lea     copying(pc),a5
        bsr     print

        clr     d0              ; Open the source file in "READ" mode
        lea     source,a0
        bsr     open_file
        bmi     bad_open

        move    d0,handle       ; Save the file handle
        move.l  dta+26,d0       ; Read this many bytes
        move.l  copy_buffer,a0  ; Into our malloc'ed buffer
        bsr     read_file
        move.l  d0,d7           ; Save status in d7 temporarily

        bsr     close_file      ; Close the file
        tst.l   d7              ; Was the read successful?
        bmi.s   bad_read        ; I guess not

        pea     dest            ; First delete the destination file
        move    #$41,-(sp)      ; (we really oughtta ask first!)
        trap    #1
        addq    #6,sp           ; We'll ignore errors from this call

        move    #0,d0           ; Try to create the destination file
        lea     dest,a0         ; with R/W status
        bsr     create_file
        bmi.s   bad_create      ; If error, bye bye

        move    d0,handle       ; Save file handle

        move.l  dta+26,d0       ; Write this many bytes (size of the file)
        move.l  copy_buffer,a0  ; From our malloc'ed buffer
        bsr     write_file
        move.l  d0,d7           ; Save status

        bsr     close_file      ; Close the file
        tst.l   d7              ; Was the write successful?
        bmi.s   bad_write       ; Nope

        lea     finished(pc),a5 ; Completion message
outta_here:
        bsr.s   print

        move.l  copy_buffer,-(sp) ; Remember to de-allocate the buffer!
```

```
        move    #$49,-(sp)      ; Mfree
        trap    #1
        addq    #6,sp

byebye:
        lea     hitakey(pc),a5  ; Prompt for keypress
        bsr.s   print
        bsr.s   getkey          ; Wait for it

        clr     -(sp)           ; We now return control of your computer
        trap    #1              ; to you

* Errors end up here

bad_open:
        lea     cantopen(pc),a5
        bra     outta_here
bad_read:
        lea     cantread(pc),a5
        bra     outta_here
bad_create:
        lea     cantcreat(pc),a5
        bra     outta_here
bad_write:
        lea     cantwrit(pc),a5
        bra     outta_here

* -------------
*  Subroutines
* -------------

* Print a line of text
*
* Enter with:
* a5 -> text to print
* device = device number
*
* Preserves a5
* Clobbers a0-a4/d0-d2

print:  dc.w    $A00A           ; Hide mouse
        moveq   #0,d0           ; Make sure this is zero
        move.l  a5,a4
print2: move.b  (a4)+,d0        ; Get character
        beq.s   p_x             ; If zero, exit
        move    d0,-(sp)        ; Print it with bconout
        move    device(pc),-(sp)
        move    #3,-(sp)
        trap    #13
        addq    #6,sp
        bra     print2          ; Keep looping til done
p_x:    dc.w    $A009           ; Show the mouse
        rts

getkey: dc.w    $A00A           ; Hide mouse
        move    #2,-(sp)        ; Get a key with bconin
        move    #2,-(sp)
        trap    #13
        addq    #4,sp
        move.l  d0,-(sp)        ; Save it
        dc.w    $A009           ; Show mouse
        move.l  (sp)+,d0        ; Get the character back
        rts

fsfirst:
        pea     dta             ; First set the dta address
        move    #$1A,-(sp)
        trap    #1
        addq    #6,sp
        move    d5,-(sp)        ; File attributes to search for
        move.l  a5,-(sp)        ; Address of search spec
        move    #$4E,-(sp)      ; Fsfirst
        trap    #1
        addq    #8,sp
        tst.l   d0
        rts

open_file:
        move    d0,-(sp)        ; Mode - 0=read, 1=write, 2=R/W
        move.l  a0,-(sp)        ; Address of filename
        move    #$3D,-(sp)      ; Fopen
        trap    #1
        addq    #8,sp
        tst.l   d0
        rts
```

```
create_file:
        move    d0,-(sp)        ; File attributes
        move.l  a0,-(sp)        ; Filename
        move    #$3C,-(sp)      ; Fcreate
        trap    #1
        addq    #8,sp
        tst.l   d0
        rts

read_file:
        move.l  a0,-(sp)        ; Address of buffer
        move.l  d0,-(sp)        ; Number of bytes to read
        move    handle,-(sp)    ; File handle
        move    #$3F,-(sp)      ; Fread
        trap    #1
        lea     12(sp),sp
        tst.l   d0
        rts

write_file:
        move.l  a0,-(sp)        ; Address of buffer
        move.l  d0,-(sp)        ; Number of bytes to write
        move    handle,-(sp)    ; File handle
        move    #$40,-(sp)      ; Fwrite
        trap    #1
        lea     12(sp),sp
        tst.l   d0
        rts

close_file:
        move    handle,-(sp)    ; File handle
        move    #$3E,-(sp)      ; Fclose
        trap    #1
        addq    #4,sp
        rts

* Construct a pathname from the output of fsel_input
* Enter with:
* a0 -> location to hold the complete pathname
* a1 -> fsel_input directory line
* a2 -> fsel_input selection line
*
* Clobbers a0-a2

make_name:
        move.b  (a1)+,(a0)+     ; Copy the directory line
        bne     make_name
make_2: cmp.b   #'\\',-(a0)     ; Back up to the backslash
        bne     make_2
        addq    #1,a0           ; First byte past the backslash
make_3: move.b  (a2)+,(a0)+     ; Copy the selection line
        bne     make_3
        rts

* Call fsel_input
* Enter with:
* a4 -> directory line
* a5 -> selection line
*
* Returns with condition code set
*   Negative=cancel
*   Zero=file selected

fsel_input:
        move    #90,contrl      ; Set the opcode for fsel_input
        clr     contrl+2        ; and the other parameters for
        move    #2,contrl+4     ; the AES contrl array
        move    #2,contrl+6
        clr     contrl+8
        move.l  a4,addrin       ; Set address of path spec for fsel_input
        move.l  a5,addrin+4     ; Set address of filename buffer
        bsr.s   aes             ; Go call fsel_input

        cmp     #1,intout+2     ; Did we click on OK? (or double-click a file?)
        bne.s   .bad            ; No, bail out
        tst.b   (a5)            ; Is there a filename selected?
        beq.s   .bad            ; No, let's scram

        moveq   #0,d0           ; Zero=good exit
        rts
.bad:   moveq   #-1,d0          ; Negative=cancel
        rts

* --------------
```

```
*  Call the AES
* --------------

aes:    move.l  #aespb,d1       ; Address of aes parameter block in d1
        move.l  #$C8,d0         ; Magic word $C8 means this is an AES call
        trap    #2
        rts

* The text strings are kept in the ".text" segment. When we don't put them
* in the ".data" segment, we can use PC relative addressing to access them.

spec:   dc.b    '\\*.*',0       ; Double backslash is necessary for MAD MAC

title:  dc.b    27,'E',27,'f',27,'p'
        dcb.b   11,' '
        dc.b    'File Copier by Charles F. Johnson (Copyright 1989 ST-Log)'
        dcb.b   12,' '
        dc.b    27,'q',13,10,10,0

src_msg:
        dc.b    27,'Y',33,32,27,'J',13,10
        dc.b    'Source file:',0
dst_msg:
        dc.b    27,'Y',33,32,27,'J',13,10
        dc.b    'Destination file:',0
cancel:
        dc.b    27,'Y',33,32,27,'J',13,10
        dc.b    ' Operation cancelled.',13,10,10,0
copying:
        dc.b    27,'Y',33,32,27,'J',13,10
        dc.b    ' Copying...',13,10,10,0
no_ram:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' Not enough RAM to copy this file!',13,10,10,0
cantshrnk:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' Not enough memory to run!',13,10,10,0
cantfind:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' Cannot find source file!',13,10,10,0
cantopen:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' Cannot open file!',13,10,10,0
cantcreat:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' Cannot create new file!',13,10,10,0
cantread:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' Cannot read file!',13,10,10,0
cantwrit:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' Cannot write file!',13,10,10,0
finished:
        dc.b    7,27,'Y',33,32,27,'J',13,10
        dc.b    ' File copied successfully!',13,10,10,0
hitakey:
        dc.b    ' Hit any key.',0

        .even

device: dc.w    2

aespb:  dc.l    contrl,global,intin,intout,addrin,addrout

        .bss
        .even

copy_buffer:    ds.l    1

contrl:         ds.w    5
intin:          ds.w    32
intout:         ds.w    32
global:
apvrsn:         ds.w    1
apcont:         ds.w    1
apid:           ds.w    1
apprvt:         ds.l    1
apptre:         ds.l    1
ap1rsv:         ds.l    1
ap2rsv:         ds.l    1
ap3rsv:         ds.l    1
ap4rsv:         ds.l    1
addrin:         ds.l    4
addrout:        ds.l    4
handle:         ds.w    1

source:         ds.b    80
dest:           ds.b    80

copy_dir:       ds.b    80
copy_file:      ds.b    16

dta:            ds.b    44

                ds.l    300
ustack:         ds.l    1
                ds.w    10

prgend:         ds.w    0

        .end
```

END

Text is entered in the same manner as a word processor except the lines will not wrap. Each entry is limited to 96 characters. The window will scroll as necessary to accommodate the entire line. Pressing Return will move the cursor to the next line and will insert a new line if the cursor is not on the last line of the file.

The level designators can be changed with Tab or Shift-Right-Arrow to move to a lower level or Shift-Left-Arrow to move to a higher level. The display is redrawn to show the effects of a level change. Once the text is entered, it can be moved via the block commands Cut and Paste. Text can be edited with the backspace and delete keys. Backspace will delete the character to the left of the cursor and move the cursor and line one character left. Delete will delete the character at the current cursor position and move the remainder of the line one character to the left.

The cursor keys or mouse are used to move the cursor within the outline. Use the right and left arrows to move to within a text line. Entering a character will insert the character at the cursor position (limited by the 96 character total line length). The up and down arrows will move the cursor up or down a field, scrolling the window if necessary to display the file. Shift-Up-Arrow will scroll the window up a page if possible, while Shift-Down-Arrow will scroll the window down a page.

A specific word can be found by using the Search function of *Outline Plus*. Press F10 to produce a dialog box for word input. Press Return and the program will search forward from the current cursor position. If the search string is found, the window will be redrawn at the word's location. Pressing F9 will renew the search for the same word. The search is case sensitive and forward seeking only.

An entire line (except for the level designator) is erased by pressing the Escape key. The entire line is deleted by pressing the Clear/Home key. The contents of the deleted line are saved to a delete buffer for retrieval if necessary (the Undo command is described below). While the text is saved, the level designator is not. The delete buffer will hold a number of records limited by system memory. The more items in the delete buffer, the fewer lines available for the edit windows.

Pressing Undo will write the last deleted item of the delete buffer to the current cursor location, allowing retrieval of an item mistakenly deleted. The delete buffer is a last-in, first-out stack arrangement. After the last item is recalled by Undo, the next to last deleted line is the next line to be undeleted via Undo. Pressing Shift-Clear-Home will clear the delete buffer and should be periodically performed after the deleted items are deemed expendable to provide the maximum number of lines for the edit windows.

To add a new line between two existing lines, press Insert or Return. Insert will add a new line at the current cursor position, pushing all subsequent lines down, while Return will insert a new line after the position of the cursor. The new line will take on the level of the current cursor line, and the window will be redrawn to reflect these changes.

Error messages

The program handles most I/O system errors and displays an alert box describing the problem. The most common are insufficient memory to load or save a file. Changing to a new disk will alleviate the problem of insufficient storage space. If the problem is insufficient computer memory, try erasing the delete buffer (Shift-Clear-Home), closing unneeded windows, and booting without any desk accessories.

Other errors are described with their GEM or TOS error code number. I have not been able to force generation of these other error messages.

Technical discussion

Outline Plus demonstrates many of the abilities of Personal Pascal, the most important being the use of linked lists to store information. The major feature of a linked list is the utilization of pointers. Pointers are variables that store the address (memory location) of a record. In addition to data, the record can store pointers that point to other records. In this manner, the records can be linked together to form a list of connected records. Figure 1 graphically illustrates a linked list and the associated pointers.

A pointer is declared in the same manner as any other Pascal type. (See Listing 1.) The information stored in this variable is the address of a record (or nothing, referred to as a nil pointer). A pointer cannot be inspected—*Writeln(PointerType)* is not legal! The syntax, *PointerType*, refers to the record information stored at the address *PointerType*. If *PointerType* is a record, then the individual items of the record are referred to as *PointerType* ∧. *StoredDataOne*, *PointerType* ∧. *StoredDataTwo*, etc. Once this relationship is understood, using pointers becomes second nature!

Recursion

The second important method utilized by *Outline Plus* is recursion. Recursion is a programming method where a procedure calls itself. Recursion makes the implementation of linked lists much easier.

In Listing 1, *AddARec* will continue to call itself until it gets to the end of the list. At that point, a new record will be created, a pointer from the previous last record will point to the newly created record (which is now the new last record), and the new information will be stored.

The basic building blocks of constructing a linked list are procedures to add a record, delete a record and insert a record (see REC_MOD.PAS and Figures 2 through 4).

Moving through the outline amounts to traversing the links between records. New lines are added via *AddARec* and *InsertARec*. Old lines are deleted with *DeleteARec*. The remainder of the program supports these procedures and displays the information on the screen.

An excellent source for more information about linked lists, pointers and recursion can be found in the text *Oh! Pascal!* by Doug Cooper and Michael Clancy.

Conclusion

Everyone who uses a word processor on his/her Atari ST can benefit from *Outline Plus*, whether one is a writer, a student or an executive. It is easy to use, versatile and dynamic. I used *Outline Plus* to help write this article, and I have never written an article faster or more comprehensively. It did not require the rewrites normally needed to add additional facts because I used *Outline Plus* to gather and correlate the facts before, not during, the writing phase.

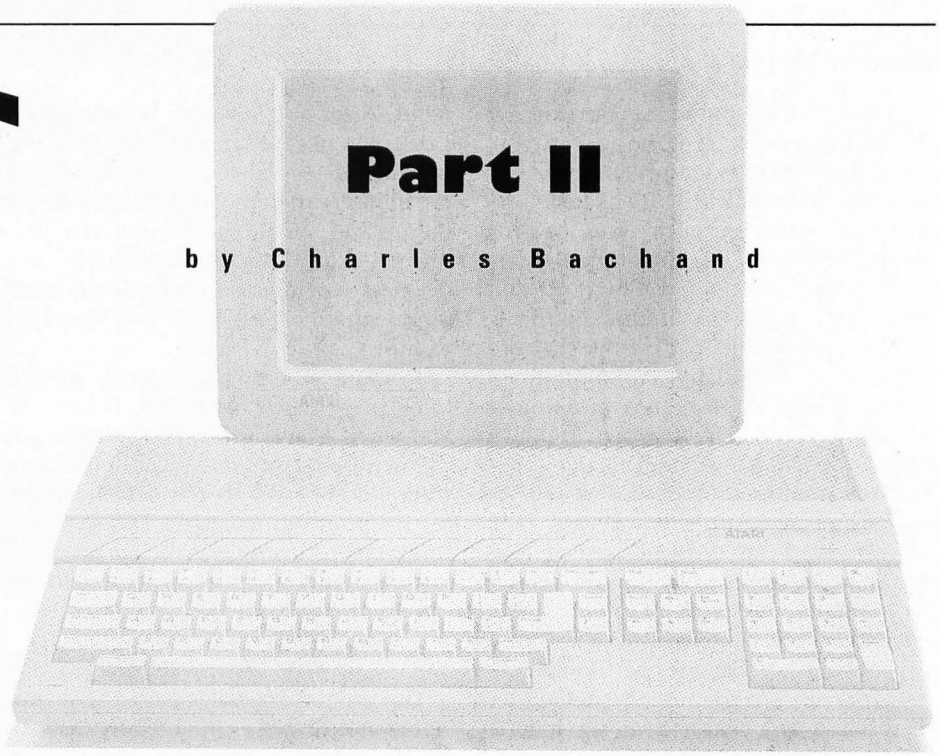
After working as a research chemist for six years, James Maki decided to become a freelance writer and make his Atari computers pay for themselves. He lives in Indianapolis with his wife, a new daughter, a golden retriever and four Siamese cats.

Mono-Gray

Part II

by Charles Bachand

HIGH RESOLUTION ONLY



Apples to Oranges

You've probably all seen an interesting piece of ST software called the *ST Xformer*. This program allows you to run Atari 8-bit programs, those destined from birth to run only on Atari XL's and XE's and their older cousins, the 400 and 800, on your ST computer. It uses software to make your 68000-based computer think it houses a 6502 processor. It does this not without some sacrifices, though.

The biggest toll is that in execution speed—the *Xformer* acts like an Atari 8-bit computer that is operating on less than half its cylinders. The job gets done, but the time it takes to do it is over twice that normally needed by a real 8-bit computer.

The slow speed of the *ST Xformer* can be attributed to the process of conversion. In this case, the software must look at every byte of the 8-bit program and ask itself, "What would I do with this if I were an 8-bit computer?" There are a lot of "what ifs" to consider, and all this pondering takes time—a lot of it. Who ever said that turning apples into oranges was easy?

We will be doing a not too dissimilar job of conversion this month when we display low-resolution DEGAS pictures using the Mono-Gray routines that were presented in our first installment.

Bit-planes, trains and automobiles

The internal makeup of an ST's monochrome screen area is quite simple. First you start with a 32,000-byte area of RAM. This memory, because of hardware constraints put on it by the graphics processor chip, must start on a page boundary (its starting address must be a multiple of 256). Each line that is displayed on the screen is made up of 80 of the total 32,000 bytes, while each bit of those 80 bytes represents one of the 640 pixels on the line. It's just simple math: eight bits per byte times 80 bytes per line times 400 scan lines equals 256,000 bits or 32,000 bytes of RAM.

With a monochrome monitor hooked up, the graphics processor chip reads and displays this 32K block of memory 70 times each and every second. It's quite simple and easy to understand. Too bad the same can't be said when working with color.

Graphics chips, and the way that data must be stored in RAM in order to work with them, are a strange lot. They were designed to show off a program's text/graphics in the shortest time possible (it's not nice to hog a computer's memory—it has other things to do). Because of this, it's not always intuitive just how graphics data is stored. Just take a look at Figures 1 and 2, and you'll see where the apples and oranges come in.

Figure 1 shows the relationship between what you see on a monochrome screen and what is found in video RAM. As you can see, there is a one-to-one cor-

respondence between the two. Figure 2 on the other hand, is an entirely different kettle of fish. Let me see if I can explain what is going on when your ST is set to its low-resolution color mode.

Let's pretend

Let's start at the beginning. You, the computer, want to display the first pixel in the top left-hand corner of the screen. Since this is the 16-color mode, we'll need four bits of information. The obvious method (that of using the first four bits in the first byte of video RAM) is, unfortunately, not the correct one. We instead must take the first bit from each of the first four words, combine them (in reverse order) before we finally get the color number. This four-bit number is an index into our array of 16 colors used to select the proper values of red, green and blue to paint at that point on the screen.

The process is repeated for the second pixel, this time using the second bits from the first four words. We continue this way until we've exhausted these (only 16 bits in a word, remember) and start all over, this time using the second set of four words. If this method of accessing the data in the screen RAM looks awkward to you, just remember that you normally have no need to handle it in this method. It is, however, the most efficient way to do it—if you're a graphics processor chip.

Fade to gray

Now let's assume that you are color-blind. You look at a color monitor, and you see but a black and white picture. You can tell the difference between two colors only if their brightnesses are different. We normally assign a value between 0 and 7 for the brightness of each of the three color guns in the picture tube. Being color-blind (remember), your mind adds these three color values together to produce a gray scale where the brightness values range from 0 to 21.

Now with all these possible brightness values to work with, you might think we should initialize the Mono-Gray routines with a value of 21, like so:

```
@ginit(21)
```

This is the ideal condition, able to handle every possible brightness level; but because twenty-one 32K screens consume 672K of RAM, most people will run out of memory before all the Mono-Gray screens are established.

There is also our old friend, "Mr. Flicker," to contend with. Cycling through 21 different screens takes $^3/_{10}$ of a second, allowing you to display only 3 ⅓ Mono-Gray screens each second. While the GFA BASIC code is optimized to reduce this flicker for colors near the middle of the gray scale, those near the ends will most likely give you a headache. No, we'd better leave trying to show 21 different shades for when we have a camera set up to do time-lapse photography.

We'd be much better off initializing for seven Mono-Gray screens and scaling down the brightness values to match our new limits of 0 to 7. Now we're dealing with only ⅓ the amount of RAM as before (now 224K), and our display rate has increased from three to ten frames per second.

Listing 1 (see DEGAS__1.GFA on disk versions of ST-LOG, as well as on DELPHI) displays low-resolution DEGAS pictures using this method. Merge it in to your Mono-Gray routines that were coded in GFA BASIC from last month at the place where you normally put your own code, then run it.

Yet another Mono-Gray routine

I should tell you at this point that I've added one more routine to the Mono-Gray code. Procedure *gframe(num&)* will allow you to view any of the Mono-Gray drawing frames individually simply by substituting a frame number for the parameter. The frame number must be an integer value in the range of *[1..gmax&]* where *gmax&* is the total number of frames. The code is comprised of only three lines, so I'm including it here:

```
PROCEDURE gframe(num&)
  VOID XBIOS(5,L:-1,L:gptr%(num&),-1)
RETURN
```

This code will allow us to easily return to viewing a Mono-Gray drawing screen after calling up a file selector box in order to load a picture file.

Seeing as how this application is being done entirely in GFA BASIC, I should warn you that the code runs a little on the slow side. GFA BASIC 3.0 happens to be one of the fastest BASICs around, but it is an interpreted (noncompiling) BASIC and so, just like *ST Xformer*, it has to identify and interpret each and every line of BASIC code it comes across. This, unfortunately, takes time. It's going to take time to draw your picture. How much, you ask? Really, not very long—only about 30 minutes.

30 minutes!

"*What*? Thirty minutes? This guy must be nuts!" you're all saying just about now. After all, it only takes Mono-Gray a split second to draw a gray square that nearly fills the entire screen. Well, the main reason for the snail's pace is that we're not using the ST's drawing routines efficiently. What we are doing is analogous to painting the side of a barn with a Q-Tip swab.

Let me try to illustrate what I mean. If we were to REM out the *@gbox()* statement (the one that calls the Mono-Gray code that actually does the drawing to the screen) the program would take only two minutes and 40 seconds to run. Why? Well, we happen to be calling a very sophisticated routine through the Mono-Gray code, the GFA BASIC statement, *BOX*.

The use of the *BOX* statement was the easiest to implement from our point of view because we happen to be drawing boxes on the screen. Unfortunately, these happen to be very small boxes (2 x 2 pixels). We also happen to be drawing 64,000 of these boxes for each of the seven Mono-Gray screens. That's a total of 448,000 squares to produce one seven-level gray scale picture!

When we use the Mono-Gray code directly to draw a large gray box that almost fills the screen via the *@gbox()* procedure, we are actually making only seven graphics calls (one for each of the Mono-Gray screens) and not 448,000 of them as is the case for DEGAS pictures. I trust that you now see why the program tends to slow down a bit.

To draw, or not to draw

The replacement *@convert* procedure in Listing 2 (see DEGAS__2.GFA) will accomplish the very same thing as that found in Listing 1, but with a substantial savings in speed. Given the task of displaying an entirely white screen in seven level gray scale, we trim our time down to 26.25 minutes—a savings of 3.75 minutes.

We gained these extra few minutes by switching to the use of the Line-A routine *PSET* that is callable from within GFA BASIC. *PSET* allows you to plot individual points with increased speed, but it will not draw boxes—or anything else for that matter. In order to draw our 2 x 2 boxes, we need to call *PSET* four times—once for each corner.

Since I didn't want to mutilate the original Mono-Gray code in the process, the Line-A routines (as well as the frame switching code found in last month's Mono-Gray procedure, *@gcode*) was substituted for the original call to *@gbox()*. This routine no longer wastes time passing parameters to a second procedure—it is now all in line code.

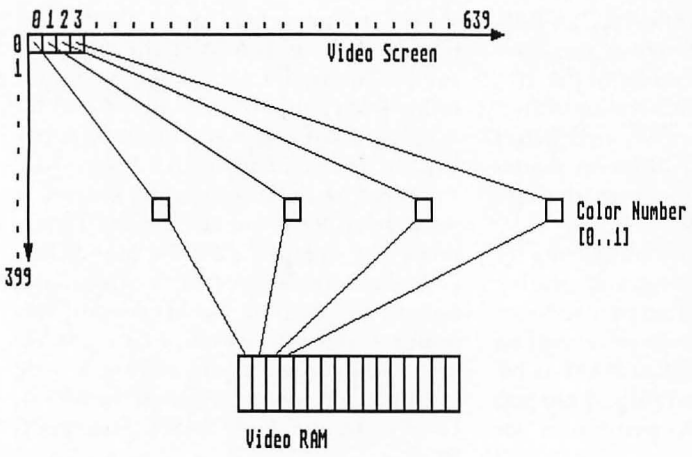


FIGURE 1 · HIGH RESOLUTION GRAPHICS

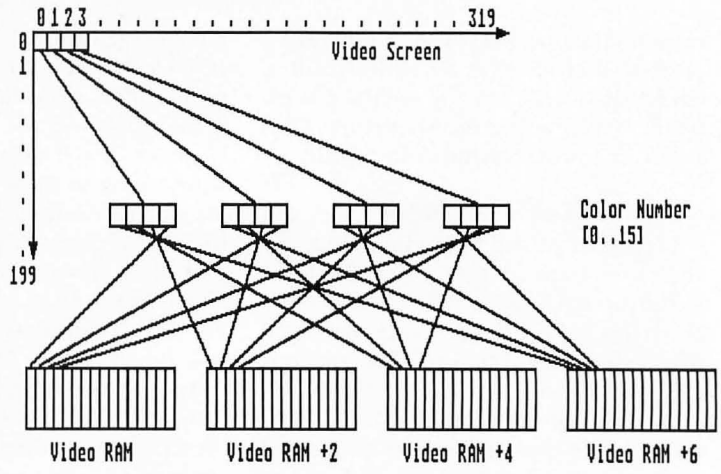


FIGURE 2· LOW RESOLUTION GRAPHICS

Incidentally, the time of 26.25 minutes we got when drawing a totally white screen happens to be this new code's "worst case" scenario, since displaying a completely black screen requires only 3.4 minutes. Why the big difference? The secret is not in what is being drawn, but rather in what is not. This new code assumes that a call to the routine *@gcls* was made before drawing began, thus filling all the Mono-Gray drawing screen with zeros. The initial setup call to *@ginit()* also erases these screens, so we don't have to worry about erasing for the first picture.

Now, since all the possible pixel positions are in their off state, we need only to draw those that need to be turned back on. In using this method, some DEGAS pictures will take longer to draw than others. On the average, your DEGAS files will require only about 15 minutes of drawing time in order to be displayed.

We no need no stinkin' graphics calls!

Remember earlier when we discovered how fast things sped up when we REMed out all calls to the graphics calls? It then took us only two minutes and 40 seconds to process a picture file. I didn't say "display a picture file," for unfortunately, nothing was drawn to the screen. Well, our final rendition of the program doesn't call these graphics routines either, but still manages to generate viewable data. How? By writing directly to the screen RAM via the use of *POKE* statements. By doing it this way, we avoid all the unnecessary code and error checking that software designed to handle three different graphics modes must contain.

Listing 3 (see DEGAS_3.GFA) takes only 3.75 minutes to generate a completely black screen. This is comparable to the 3.4 minutes taken up by Listing 2 to perform the same feat. Handling a totally white screen (and anything in between) is where the code in Listing 3 really shines. Listing 2 took 26.25 minutes to do this—Listing 3 does it in 7.5!

To do this, we need to do all the masking, shifting and poking to screen memory ourselves. One might think that since it is being done in BASIC that it would be actually slower than if the operating system were handling the graphics. But our particular application actually runs faster in BASIC.

The main trick is in the substitution of the following line of GfA BASIC code for the call to *@gbox()* in Listing 1 or the four calls to the Line-A routine *PSET* in Listing 2:

```
video%(index&)=OR(video%(index&),
    SHR(&HC0000000,i&+i&))
```

This line of code does basically what the graphics calls before it had done, but it does them to an array of seven longword variables called *video%()*. It builds up the 32-bit wide values by ORing them with a two-bit wide mask (&HC0000000) that gets shifted two bits to the right for each pixel. We're using two-bit masks and two-bit shifts because we want to plot two pixels in width for each pixel in the color DEGAS picture. The actual plotting is being done by the following code:

```
FOR index&=1 TO gmax&
  LPOKE gptr%(index&)+offset&,video%(index&)
  LPOKE gptr%(index&)+offset&+80,video%(index&)
NEXT index&
```

Now since we also want to double up the pixels in the vertical direction to properly display a 200-line picture on a 400-line display, we need to include the second *LPOKE* statement with the value of 80 added to the screen offset. Since there are 80 bytes of data on each scan line, this will display our second longword right under the first. We now need to position the offset value four bytes further along the display with:

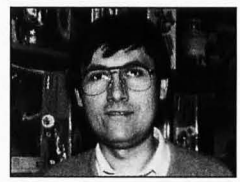
```
ADD offset&,4
```

Now we need a way to skip over all the odd-numbered scan lines (odd if numbered from 0..399) so that the first *LPOKE* will not wipe out data left by a previous use of the second *LPOKE*. That's where the use of the following *IF* statement comes in handy:

```
IF MOD(offset&,80)=0
  ADD offset&,80
ENDIF
```

This says that if the new offset value we got after adding 4 to it is evenly divisible by 80 (which is the case at the beginning of each scan line), then add 80 more bytes to it so that it points to the beginning of the next scan line.

Well, that's all the room we have for Mono-Gray in this issue. Next time we'll learn how to display art that has been compressed into the Tiny format, and we'll see if we can speed up things still further using machine language subroutines. Until then, may your blue skies be gray.



Charles Bachand, when not tooling around town in his 300ZX, can usually be found racing R/C cars or busily managing his own area on DELPHI, the Hobby SIG. His username is, appropriately, BACHAND.

MONO-GRAY Listing 1: GFA BASIC 3.0

```
' DEGAS picture displayer with eight levels of gray [0..7]
' Listing 1. by Charles Bachand for ST-Log Magazine
'
' The first and "slowest" version of the software.
'
DIM palette&(15),work&(15),picture&(16000)
@ginit(7)
DIM video%(gmax&)
findpic:
@gexit  ! go to original screen for file selector box
SETCOLOR 0,1
FILESELECT "*.PI1","",fname$
IF fname$=""
  END
ENDIF
OPEN "I",#1,fname$
a&=INP(#1)+INP(#1)
IF a&<>0
  ALERT 3,"Not a low-res|DEGAS pic!",1,"Sorry",a&
  GOTO findpic
ENDIF
FOR index&=0 TO 15
  palette&(index&)=INP(#1)
  a&=INP(#1)
  ADD palette&(index&),(a& AND &HF)+SHR(a&,4)
NEXT index&
BGET #1,VARPTR(picture&(0)),32000
CLOSE #1
@gframe(1) ! show first frame while drawing
SETCOLOR 0,0
@convert
@gshow
GOTO findpic
'
PROCEDURE convert
  LOCAL i&,pntr&,xptr&,yptr&
  pntr&=0
  xptr&=0
  yptr&=0
  REPEAT
    ARRAYFILL work&(),0
    FOR i&=0 TO 15
      IF BTST(picture&(pntr&),i&)
        INC work&(15-i&)
      ENDIF
      IF BTST(picture&(pntr&+1),i&)
        ADD work&(15-i&),2
      ENDIF
      IF BTST(picture&(pntr&+2),i&)
        ADD work&(15-i&),4
      ENDIF
      IF BTST(picture&(pntr&+3),i&)
        ADD work&(15-i&),8
      ENDIF
    NEXT i&
    ADD pntr&,4
    FOR i&=0 TO 15
      gcolor&=(palette&(work&(i&))+2)/3
      @gbox(xptr&*2,yptr&*2,xptr&*2+1,yptr&*2+1)
      INC xptr&
      IF xptr&>319
        xptr&=0
        INC yptr&
      ENDIF
    NEXT i&
  UNTIL yptr&>199
RETURN
```

MONO-GRAY Listing 2: GFA BASIC 3.0

```
'
' This faster version of the convert procedure
' draws only the White areas of the screen.
'
PROCEDURE convert
  LOCAL i&,pntr&,xptr&,yptr&
  pntr&=0
  xptr&=0
  yptr&=0
  REPEAT
    ARRAYFILL work&(),0
    FOR i&=0 TO 15
      IF BTST(picture&(pntr&),i&)
        INC work&(15-i&)
      ENDIF
      IF BTST(picture&(pntr&+1),i&)
        ADD work&(15-i&),2
      ENDIF
      IF BTST(picture&(pntr&+2),i&)
        ADD work&(15-i&),4
      ENDIF
      IF BTST(picture&(pntr&+3),i&)
        ADD work&(15-i&),8
      ENDIF
    NEXT i&
    ADD pntr&,4
    FOR i&=0 TO 15
      gcolor&=(palette&(work&(i&))+2)/3
      gcounter&=0
      FOR index&=1 TO gmax&
        SUB gcounter&,gcolor&
        IF gcounter&<0
          ADD gcounter&,gmax&
          VOID XBIOS(5,L:gptr%(index&),L:-1,-1)
          PSET xptr&+xptr&,yptr&+yptr&,1
          PSET xptr&+xptr&+1,yptr&+yptr&,1
          PSET xptr&+xptr&,yptr&+yptr&+1,1
          PSET xptr&+xptr&+1,yptr&+yptr&+1,1
        ENDIF
      NEXT index&
      INC xptr&
      IF xptr&>319
        xptr&=0
        INC yptr&
      ENDIF
    NEXT i&
  UNTIL yptr&>199
RETURN
```

MONO-GRAY Listing 3: GFA BASIC 3.0

```
'
' This convert procedure POKES directly into screen RAM.
'
PROCEDURE convert
  LOCAL i&,offset&,pntr&
  offset&=0
  pntr&=0
  REPEAT
    ARRAYFILL work&(),0
    FOR i&=0 TO 15
      IF BTST(picture&(pntr&),i&)
        INC work&(15-i&)
      ENDIF
      IF BTST(picture&(pntr&+1),i&)
        ADD work&(15-i&),2
      ENDIF
      IF BTST(picture&(pntr&+2),i&)
        ADD work&(15-i&),4
      ENDIF
      IF BTST(picture&(pntr&+3),i&)
        ADD work&(15-i&),8
      ENDIF
    NEXT i&
    ADD pntr&,4
    ARRAYFILL video%(),0
    FOR i&=0 TO 15
      gcolor&=palette&(work&(i&))
      gcounter&=0
      FOR index&=1 TO gmax&
        SUB gcounter&,gcolor&
        IF gcounter&<0
          ADD gcounter&,gmax&
          video%(index&)=OR(video%(index&),SHR(&HC0000000,i&+i&))
        ENDIF
      NEXT index&
    NEXT i&
    FOR index&=1 TO gmax&
      LPOKE gptr%(index&)+offset&,video%(index&)
      LPOKE gptr%(index&)+offset&+80,video%(index&)
    NEXT index&
    ADD offset&,4
    IF MOD(offset&,80)=0
      ADD offset&,80
    ENDIF
  UNTIL offset&>=32000
RETURN
```

END

OUTLINE PLUS · PROGRAM LISTINGS

(From article beginning on page 40)

OUTLINE PLUS
Listing 1:
Pascal

```
Type
   RecordPointer = ^ExampleRecord ;    { Notice that the Pointer
                                         Type is actually declared
                                         BEFORE the record type }
   ExampleRecord = record
                   { Data to be stored }
                    Data       : integer ;
                   { Pointer to next record }
                    NextRecord : RecordPointer ;
                   end ;


   procedure AddARec(Var CurrentRecord : RecordPointer ;
                         NewData : integer ) ;

      begin
        if CurrentRecord = nil then
           begin
             New(CurrentRecord) ;
             CurrentRecord^.Data := NewData ;
             if FirstRecord = nil then
                FirstRecord := CurrentRecord ;
             LastRecord := CurrentRecord ;
           end
        else
           AddARec(CurrentRecord^.NextRecord, NewData) ;
      end ;
```

END

REVOLUTIONARY NEW PRODUCT

SWITCH BACK

REQUIRES at least 1 meg. of RAM
(or a Megadisk or Polydisk Cartridge)

- Imagine Saving almost any game at any point, then being able to return there as many times as you like.
- Imagine the Ultimate Back-up Utility that actually UNPROTECTS programs as it copies them. Lets protected programs be stored as files, run from a hard disk or even be transmitted over a modem.
- Imagine saving three or more protected single sided disks on just one double sided disk.
- Imagine Instantly switching back and forth between two different programs, games, utilities or business applications.

Now Stop Imagining and get Switch/Back.
It can do all this and more.

Switch/Back is a revolutionary new hardware and software package that lets you get more from your ST, MUCH MORE.

Switch/Backs gaming features lets you instantly save most games then continue playing. If you get in trouble you can switch back to where you were as many times as you like.

BACK-UPS –Switch/Back can work with your favorite back-up program and allow you to save whole protected disks to files for archival purposes. It can also automatically unprotect a program and save it as standard file. This method works on hundreds of ST programs and it allows you to run the files directly. Its perfect for running protected programs off a hard disk. It creates standard TOS files, that can be stored together on disks or even transferred by modem.

SWAP – Switch back lets you load just about any two programs into your ST and switch instantly between them. It works with games, business programs, utilities, compilers, etc. Although only one program is running at a time, the other is available instantly, right where you left off.

The Switch/Back hardware plugs into your printer port for easy use (It has a pass through connection for your printer too.)

Switch/Back requires at least One Meg of memory (Or a Polydisk or Megadisk)

ONLY $69.95

ST Protection Techniques

Finally ST Copy protection techniques are revealed. This complete book and disk package details the state of the art in ST Protection methods and much, much more.

The Software included with the book provides many powerful features like the AUTOMATIC PROGRAM PROTECTOR. This easy to use Utility allows you to protect just about any ST program. You can choose a combination of protection methods like encryption, checking custom disk formats, password protection or a limited use option that makes the program self-destruct after running a preset number of times.

The book includes topics such as Phreaking, Logic Bombs, Hardware data keys, the legal aspects of piracy and software protection, Custom disk formats, Pirate Bulletin boards and much more.

In addition it contains reviews of the popular ST back-up programs and detailed explanations of ST disks and drives.

ST Protection Techniques (Book and disk package) **Only $39.95**

Digisound ST

High Quality sound digitizer for the ST. This powerful hardware and software package lets you sample real world sounds and play them back on any Atari ST. Add special effects like Echo, Reverse, looping, pitch manipulation, mixing and envelope control. Turns your Atari keyboard into a musical instrument to play songs with your digitized sounds (also works with any MIDI keyboard). Digisound makes it simple to add sound to your own program, too! Unleash the incredible sounds in your ST with DIGISOUND. Supports sampling from 5 to 40Khz, DIGISOUND is the choice of the professionals. DIGISOUND was used to create the voice in Chessmaster 2000, and other commercial programs. **ONLY $89.95**

DIGISOUND PROFESSIONAL

All the excellent features of DIGISOUND plus these great extras

LOGARITHMIC SAMPLING – Special hardware extends the sound quality far above the other ST sound digitizers. Logarithmic sampling and playback (external amplifiers only) greatly extends the dynamic range while reducing distortion and noise.

Internal Real Time Mixing – Input from a stereo and a microphone so you can sing over a tape.

$149.95

Beat Box

Is it a Drum Machine? A sequencer? A new concept in digital sound? The answer is - YES!! It's all this - and so much more!! It's a polyphonic song construction set that turns your ST into a drum machine and digital sequencer. Now anyone can be a master composer. No musical knowlege required!

Just point and click to create facinating drum, volce, or musical patterns in four voices. Combine and arrange patterns to form complete musical compositions. Play the sounds, patterns and songs through your montior speaker or digitizer hardware.

You don't need a digitizer to enjoy Beat Box. It comes with over 35 ready to use digitized sounds. Or you can use your own sounds recorded with a Digisound ST, Professional, or other digitizer.

Beat Box $29.95

COLOR COMPUTEREYES™

Incredible COLOR video digitizer. • The first and only full color digitizer for the ST. • Uses standard video inputs like video camera, VCR, or video disk. • Works in all ST resolutions, Low res provides 16 shade black and white or full color pictures. • Pictures can be used with Degas, Neochrome, Powerprint and others. • Automatic calibration of contrast, brightness and white balance. • Plugs into cartridge port for easy set-up. • Capture your picture or that of your favorite star. **ONLY $199.95**

SPECIAL OFFER – Buy both Computereyes and Powerprint and SAVE 20.00 from the total.

PowerPrint ST

BLOW YOURSELF UP

Imagine your picture on a 6 foot poster. Create a business graph that can cover a wall. Quality output for posters, t-shirts, news letters, and more.

POWERPRINT

Whether it's a photo digitized with ComputerEyes, a masterpiece created with Degas, or the winning screen from your favorite game, POWERPRINT can print it with unequaled clarity and resolution. PowerPrint supports **ALL** ST resolutions. It prints multiple sizes up to **GIANT WALL SIZED POSTERS.** Print 16 shades for incredible detail. Print the whole screen or **ZOOM** in on just the part you want. POWERPRINT offers unique effects, including rotate, mirror and inverse options. Selective shading option allows you to print multi-color pictures on any printer by printing one color at a time (using color ribbons). Powerprint lets you capture and print almost any ST screen. Works with Star, NEC, Citoh, Gemini, EPSON, XM8048 and compatible printers.

Only $39.95

DIGISPEC

DIGISPEC is an exciting new breakthrough in computer video digitizing. **DIGISPEC** works with your Color ComputerEyes to create **spectacular 512 color** video images. Now you can capture and display video pictures in unsurpassed detail and clarity, with 512 different colors on your ST screen.

DIGISPEC is easy to use. It works on any ST computer. Simply capture an image with your Color ComputerEyes and color video camera (or VCR, video disk, etc.). Then run **DIGISPEC,** and watch as your picture appears in a full rainbow of 512 true colors.

DIGISPEC includes a special shading feature to give you even more detail and color. The shading feature uses a technique called dithering, which creates and displays images in 3,375 or even 24,389 simulated colors.

DIGISPEC creates detailed, low resolution video images on any Atari ST, color video camera (or VCR), and Color ComputerEyes. Every Color ComputerEyes needs **DIGISPEC!**

Only $39.95

Polydisk

Polydisk is a 512K version of a Megadisk. Polydisk gives you the same fast boot features, the high speed access, and the print spooler. Polydisk has a power supply (like Megadisk) but does not contain a battery back-up.

Note: Those with only 512K of main memory can use Switch/Back with a Polydisk, just like those with one Meg.

Polydisk (512K Solid state drive)
Clock option card is also available for Polydisk

24 HOUR HOTLINE — VISA & MasterCard Welcome

216-374-7469
Call or write for free catalog.
Customer Service line (216) 467-5665 M-F 9 AM-3 PM EST

ALPHA SYSTEMS

Order by phone or send check or money order to:
ALPHA SYSTEMS 1012 Skyland, Macedonia, OH 44056
Include $3.00 shp. & hdlg. (US & Canada). Ohio residents add 5½% sales tax. Foreign orders add $8.00

CIRCLE #108 ON READER SERVICE CARD.

Art & Film Director

Epyx Inc.
600 Galveston Dr.
Redwood City, CA 94063
$79.95, color only

Reviewed by Maurice Molyneaux

When I decide to review a program, I usually do it within a few weeks of receiving it. Thus, it's a little odd to be reviewing a package that I've used for such a long time. Oh, the software in question only shipped about three months ago, but as of this date I've been using various versions of it for some *two years* (as a tester)! Therefore, I feel pretty well qualified to review the package.

I am referring to Epyx's *Art & Film Director*, which contains two independent, but interrelated programs. *Art Director* is a powerful paint program used for creating pictures and/or backgrounds and material for *Film Director*, the animation program that accompanies it. I'll refer to each program as *Art* or *Film* from here on.

A quick aside: Separate versions of each of the programs in this package are marketed in Europe, but these are not the same versions sold here. Epyx's versions are more powerful and redesigned. All versions of the software were programmed in Budapest, Hungary by programmers at Novotrade Software.

Art Director

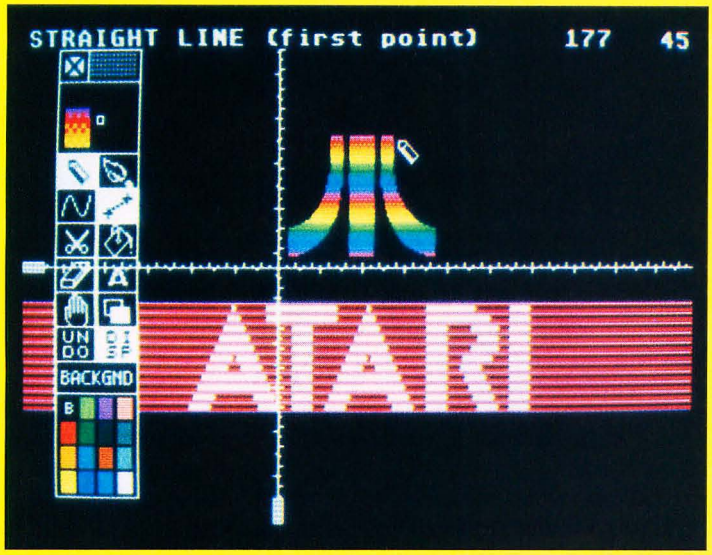
Let's look at *Art* first. At a glance its list of features looks similar to those of *DEGAS Elite* (*DE*). True, there's a lot in common, but *Art* goes more than a bit beyond *DE* in a number of important ways. First, at maximum *DE* allows you eight workscreens; *Art* allows 16 (two if you have only 512K). To make these multiple screens more useful, *Art* contains a number of tools for copying pictures, palettes, pixels, etc., from one page to another. One mode lets you "scrape" away part of one page to see through to another. *Art* also allows you to treat two pages like one canvas, scrolling to work on the junction between them. (You can also print both as one image.) There is even a function that allows you to slide the current page around the screen, making whatever disappears off one side of the screen reappear on the opposite side!

As for color, while *Art* does not allow more than 16 colors at any one time (like *Spectrum 512* or *Quantum Paint*), you are not trapped to only 16 for an entire picture all the time. For every picture you can define up to eight separate color palettes, each of 16 colors, and then cycle them on the screen. In this way you are not just rotating the color values into different positions on the palette (as in *DE*), but can actually change the colors. If you wish to print a picture to a standard printer, there is even a "mono" view mode that will show the current picture in gray scales, allowing you to get a preview of how it will print.

Aside from the usual tools for drawing freehand, making geometric shapes and the ability to copy chunks of the screen to stamp down, draw with or resize/distort, there are a plethora of tools not found in other ST paint programs. For example, the major difference in *Art* is that when you create a "brush" (the equivalent of *DE*'s block), you can use it with almost any tool, including, but not limited to, such functions as circles, lines, airbrushing, stippling, etc. One major point is that you can use any brush as a fill pattern, no matter what its size. This is in contrast to *DE*, which limits you to fill patterns that are exactly 16 x 16 pixels in size. You can cut out rectangular, oval and irregular (freehand) brushes and draw, paint or fill with them. Fortunately, *Art*'s brushes, even when they're as big as the entire screen, don't blink as annoyingly as they do in *DE*.

One additional nicety about *Art*'s fill is that you can toggle it to a mode called "fill contour," which will fill with the current color or brush, overwriting everything that is not the same color as the one currently selected in the palette. In this way you can fill over multiple colors or even other fill patterns!

There are many tools for manipulating parts of your pictures. For example you can rescale, stretch and rotate images, in addition to bend them to look like a



half cylinder, distort them by repositioning the four corners of the source fragment or even bulge a circular image so that appears as if viewed through either a convex or concave fish-eye lens (you determine the severity of the effect). There is even a perspective mode that allows you to select a vanishing point and will then plot any source graphics fragment in perspective relative to that point. For example, to draw the cobblestone for the cover of last February's *VideoGames & Computer Entertainment*, I drew an overhead view of a patch of cobblestone, then used perspective to lay out the pieces on a grid-like tile! *Art* managed all the diminishing sizes for me.

Furthermore, *Art* provides tools for lightening and darkening colors locally (this replaces the selected color with another color in the palette that is the closest thing to a lighter or darker shade of the color), meaning you select the area where the shade changing will occur. Furthermore, there are tools for replacing one color with another or swapping any two colors. There are also functions for pulling all the pixels of a selected color from one picture to another, allowing you to combine images in interesting ways. You can also outline all areas of one color with a borderline of another color. And, in addition to these being global effects (which work on your entire picture), you can limit the area of the picture they will effect by using a special window mode.

Fonts are not the usual ST or GDOS fonts, but actually screens of text that you load on one page and can type with on another. Thus, you can have letters from normal size all the way up to a huge 32 x 48 pixels, in colors from a single one all the way up to the system limit of 16. Furthermore, by using these "font pages" for other graphics, you could make libraries of clip art you could then "type" on other pictures.

Art can tell a font file by certain bits that are set in its header. If you copied a font page to another screen, modified it and saved it, the program would not recognize it as a font file when you reloaded it, because the screen you saved it from did not contain the font-definition information. This is an annoyance, and you should be able to force the program to treat the screen as a font of a given size if you so desire.

The program even features odd little functions for ping-ponging (sprite) a small part of one page atop another and another feature for spinning part of a picture as if it were printed on a card. An interesting animation function is also supported using this spin mode, but unfortunately Epyx did not include an appropriate example page with the program (the file in question has to have certain bits set to tell the program what it is for), nor is this function hinted at in the manual.

One of the main uses for *Art* is to create graphic elements that are collected on screens/pages, often called "shape tables," which are saved to disk and then imported into *Film Director* for animation purposes.

Film Director

Film is an animation program, but one with little similarity to something like *Cyber Paint*. It is a "cel" animation system, which refers to the fact that various images are treated as if they were stuck on sheets of transparent material, known as a cel in the animation biz, which can be placed over other cels and/or backgrounds. These cels can be arranged into predefined "loops" (repeating sequences) of animation or manipulated individually over a series of frames to create the movements necessary to create the illusion of movement.

Film uses two basic building blocks: the polygon and the pattern. A polygon is just a closed, unfilled geometric shape composed of lines (each of which can be a different color). Unlike those of *Aegis Animator ST*, *Film*'s polygons are not dealt with on a frame-by-frame level and are neither as powerful or as useful. Then again, they are not the main function of *Film*, as they are in *Animator ST*. The primary unit is called a pattern, which is a block of graphic data you clip from a pattern page (usually one drawn in *Art*).

A page might contain various positions of a puppy's body and a separate set of heads that you "cut" out to make patterns. You can assemble multiple patterns and/or polygons to make more complex elements, such as stages (backgrounds), groups (multiple items treated as one object) and actors (a sequence of groups/cels treated that can be automatically recorded).

To make things a little easier, the program can automatically generate in-between forms for polygons (you define one as the start, a second as the end, and tell *Film* how many intermediate forms you want), in addition to positioning and selecting cels in groups, stages, frames, etc. There is even the ability to interpolate in-betweens on existing frames without effecting the animation already defined.

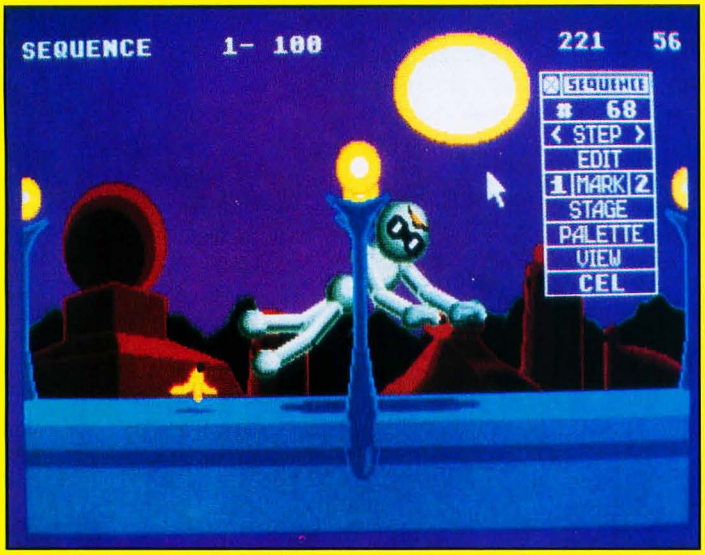
If it sounds complicated, don't worry, it is! But the complexity shouldn't scare you off. The reason that the program is complex is because it's powerful. *Film* contains seven different editors for defining patterns, polygons, groups, actors, frames, sequences and color palettes. This might sound extreme, but it allows you complete control over your animation. Changes in one editor are immediately reflected in another. If you delete a pattern from the pattern editor, that pattern will vanish from any group, frame, etc., it appeared in. This is very handy. Furthermore, because of the way *Film* stores and plays its animation, it is possible to stop in the middle of it, change something and then run it again.

The sequence editor is one of the most useful. It allows you to define a starting and ending frame and carry out certain operations on all frames within that range. For example, you could change the stage or palette on a series of frames, turn on the trace mode (which does not erase the previous image before drawing the next, leaving a trail of afterimages), overlay the cels from the range onto other frames, or even peel the top layer of cels off of every frame in the sequence!

Film also contains its own simple drawing tools in the pattern editor, allowing you to add items or edit them without having to save your work and return to *Art*. This can save a lot of time.

It is difficult to describe a program like *Film* adequately. As the old saying goes, actions speak louder than words. What I found *Film* particularly good at was animating cartoon characters and creating flat, stylized animations. The main advantage is that you can make minute manipulations all the time, adjusting elements, all without redrawing (you just replace and/or reposition cels), al-

FILM DIRECTOR



lowing you to refine animations endlessly. By building sufficiently complex shape tables, you can make detailed and flexible characters (this is not as easy if you only have 512K).

Film cannot be said to compete against the *Cyber Studio* (*CAD-3D*) as the animations it generates are of a different ilk entirely. However the type of animation it was designed to do, it does very well—and better than you could do with any other ST animation program.

The programs come with a small utility for converting *DEGAS* (it claims "Elite," but it will not convert compressed *DE* files), *Neochrome* and *Art* format files from any format to another. While this is nice, it only converts one file at a time and doesn't automatically remember the source and destination paths for the conversion when copying multiple files.

One small annoyance is that *Film*'s animation files are in a format that cannot be ported to other formats. It would be nice to see a utility to convert a *Film* file to something like a *Cyber Paint* delta or sequence file. Both also come with stand-alone player programs that can be freely distributed. Unfortunately, the ARTSHOW slide show program must be in the same directory as the pictures to be shown, as you cannot select a path from which to show pictures.

In general

Both *Art* and *Film* use a mouse-driven GEM-like user interface, with drop-down menus and dialog boxes. One nice touch is that the menu bar does not appear until you want it to, which allows you to see your whole picture at once. You can set the mouse pointer so it will be restricted to the visible screen or will be able to move beyond its edges (useful for dragging items partially off screen), and toggle on or off warning messages.

Both programs have toolboxes for selecting common tools, and these toolboxes can be moved horizontally across the screen or removed entirely. I particularly like the toolbox in *Art*, because you can select a tool by left-clicking on its icon. Right-clicking on the same icon gives you access to a box that allows you to select the settings and options for the tool. Very nice.

Both programs allow you to toggle on status lines that report the current tool (often explaining which step you are on) and the coordinates of the mouse pointer (for fine positioning). Also, in both programs you can turn on a set of "rulers" that appear along the screen edges. The manual fails to note that these rulers can be moved by clicking on and holding down the left mouse button while pointing at the small block at the root of each ruler. You can therefore use the rulers for lining up elements in your drawing or while animating.

Another nice feature is that just about every function in both programs can be called into action with a keyboard command. I heartily recommend learning these as it speeds up your work by severalfold in the long run. Oddly enough, for programs that are so mouse-dependent, you are forced to type either a *Y* or an N to confirm quitting the program. This is inconsistent.

Now we get to some of the problems with the programs. First and foremost, both work in low resolution only. You can't even run them from medium resolution (which is a simple thing to do).

Strangely, for two programs so closely linked, there is no way to jump from one to another. To go from *Art* to *Film* or vice-versa you must quit to the desktop and then run the other program. The programmers should have used GEM's *shel_write* call to have the desktop automatically run the other program when the current one exited. (It would have been easy for it to have gone to a specified directory path for this.)

Worse than these other problems, both programs (along with their stand-alone "player" programs) are hard addressed, meaning they are designed to load at specific places in your ST's memory. Thus if something else is residing there, like a desk accessory or program you ran from your AUTO folder (like *Templemon* or *UIS II*), when you try to run *Art* or *Film*, the programs will not load! The only way I have discovered to circumvent this is to have a machine with over 1024K (I have a Mega ST4) and use CodeHead's *TopDown* loader (which forces accessories and AUTO programs to load at the top of RAM, out of *Art*/*Film*'s way). This solution doesn't seem viable on machines with one meg or less RAM.

Another problem is that *Art* and *Film* do something slightly wrong when calling up GEM alert boxes and menu bars. Occasionally you will hear a sound like a key being held down. It only lasts an instant, then stops. The reason it stops is that the programmers shut off the ST's key repeat, so the programs have no key repeat at all. If you want to move your mouse pointer using the arrow keys, you'll have to tap them over and over! (One way around this is to call up the *Templemon* monitor (if you have it—it's public domain) from inside one of the programs and switch the key repeat on by using the command ":484 7" [sans quotes]).

The manual is about 150 pages long, and pretty good, all in all. Unfortunately, some details have been missed (such as the aforementioned fact that you can move the rulers), and there are some flat-out errors. The keyboard command guide for *Art* omits some information regarding that you must already be in a specific mode before the commands listed will work. Also, the section on videotaping your animation mentions only STs equipped with composite video output jacks—which just don't exist!

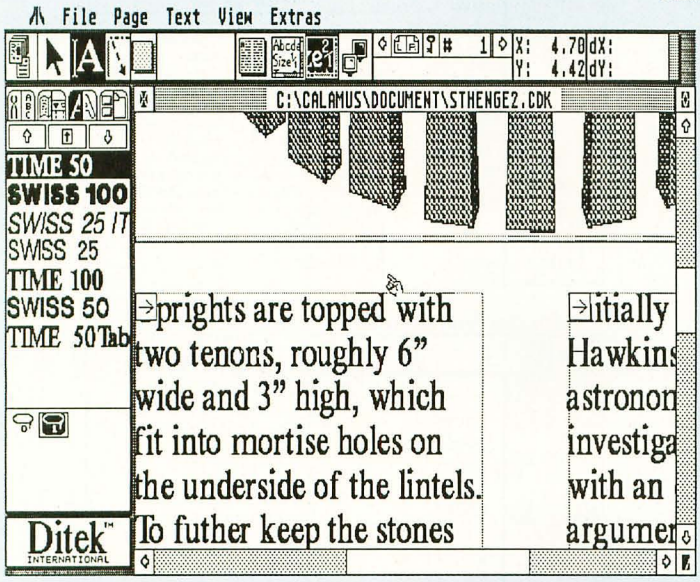
Big bothers all, but not insurmountable ones. Interestingly, this package marks Epyx's first step into applications software on the ST. If the product does not go over well, it may also be its last, which would be too bad. As of this date the programs have been available for over three months, and Epyx hasn't advertised them yet. Perhaps it doesn't realize that most ST users won't buy applications software if they don't know what it is.

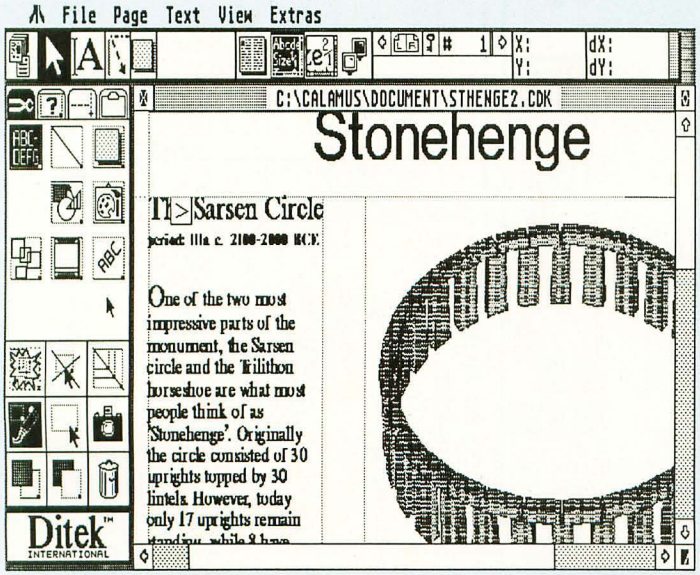
Finally, should you buy this package? Well, if you are a serious graphics enthusiast and particularly if you are interested in animation, I'd say go ahead. You can do some amazing stuff with this software if you try. I've created game graphics and fully animated cartoons with it. Furthermore, you're getting two programs (four single-sided disks) for $79.95 retail, which isn't a bad deal at all. Even better though is the fact that this package is often discounted to about $50 or slightly less. Good stuff—if you can live with its idiosyncracies. ■

Calamus

ISD Marketing Inc.
2651 John Street, Unit #3
Markham, Ontario
Canada L3R 2W5
(416) 479-1880
$299.95
Requires one megabyte of RAM, double-sided disk drive and monochrome monitor.

Reviewed by Ian Chadwick





Prepare to be impressed.

***Calamus* brings a whole new level of** desktop publishing to the Atari ST, reaching the professional user, but at a price still affordable by the serious amateur. Compare this to the $895 suggested list price for *Ventura Publisher 2.0* (plus $595 for the Professional Extension package), and you'll know what I mean.

Calamus is a mixture of power and grace, providing a wealth of features that rival and often surpass the professional-level PC desktop publishing packages—with oddities and awkwardness. Unfortunately, *Calamus* contains flaws, many of which are minor and appear to be simple oversights. The programmers aiming at the big picture have failed to make sure that all of the small elements were taken care of. Many problems arise from the program's design and user interface.

The sheer number of features and functions in this program, coupled with the enhanced user interface, make *Calamus* far more demanding and difficult to use than either *Publishing Partner* or *Timeworks' Desktop Publisher*. On the plus side, the control over the output and the ability to manipulate the page contents and the display is correspondingly higher. But be prepared to spend frustrating hours pouring over the manual looking for explanations and hints as to how features actually perform. Without an index to guide you, this is an unrewarding task and—even when a reference is found—the result may be disappointing or unexpected. The manual is far too unrevealing, forcing the user to experiment constantly to discover the limits of almost every command.

For example, there is an icon titled "header/footer frame." By this, one assumes you can create both types of frames: headers *and* footers (referred to as "headlines" and "footlines" in the manual). This is not contradicted in the manual. But when you try to create both types, you discover you can have only "one header/footer frame on a page," as the error message warns you. Some experimentation shows that several frames can be joined into a single header/footer group.

Some features prove better than you expect. The display is truly WYSIWYG—something *Ventura* lacks. *Calamus* displays all fonts in crisp clarity at all sizes. This is due to the use of vector (not raster) fonts. No matter what size a font is magnified or reduced to, it remains highly visible. *Calamus* can also display fonts in rotated text boxes in any angle, even upside down!

While *Calamus* comes with only two font families (a serif and a sans serif in several sizes and italic faces), ISD signed a deal with Compugraphic to sell 96 of its fonts to registered users. Since *Calamus* can talk directly to a Linotype machine (a raster image processor, RIP, is under development and should be for sale soon),

this really moves *Calamus* into the professional DTP league. For those of us with more modest needs, a vector font editor package is due for release in the first quarter of 1989.

One of the nightmares a layout designer must work with is the shifting nature of available space in a commercial publication. Articles, if not edited to fit the space, often have to jump great gaps of pages to continue. *Calamus* handles this with ease, even permitting jumps to previous pages, not merely those following. It also supports automatic text flow from one page to another.

More than a simple layout program, *Calamus* includes a host of graphic design elements such as raster frames in a wide variety of shades, backgrounds, shadowing and shapes, half-tone picture sizing, optimum picture size for screen and printer, cut pictures that show any portion of the image, numerous grid, margin and help-line functions, rulers, a five-part clipboard that accepts entire frames, text and even graphics, 15 line types in background pattern, shade, end or thickness and more.

Calamus automatically handles kerning in all fonts and at all sizes. There are additional controls to move characters and adjust word, character and line spacing.

In text manipulation and control, *Calamus* only provides lukewarm support. The text editor is so embarrassingly primitive as to be barely deserving of the name "editor." It should be ignored except where absolutely unavoidable. The only redeeming characteristic it provides is that text appears formatted in the same width as the frame it represents.

The "index" function is misleading and worthless: It can neither sort entries alphabetically nor provide page numbers, see or see-also references for the entries! It merely copies or moves highlighted text to a specific frame, in the order it finds them in the original frame. On the plus side, you can control such elements as page and chapter numbering, hyphenate text, insert footnotes and control text style, size and display type to an amazing degree.

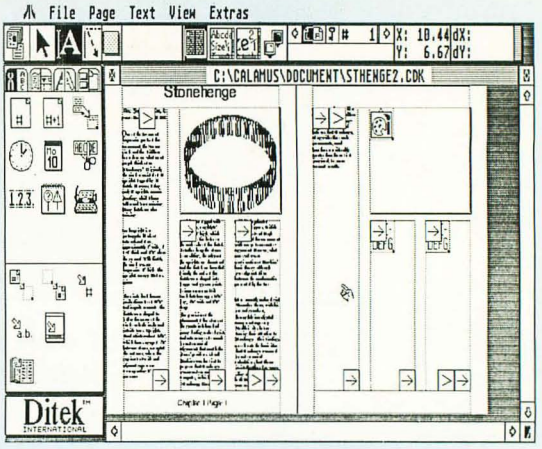
You can import or export text as ASCII, *Calamus*, *MS-Write* and *1st Word* formats, but not *WordPerfect* or *ST Writer*. Raster graphics files can come in as *DEGAS* (files and blocks), *Calamus*, GEM Image (.IMG), GFA BASIC blocks, IFF and STAD (a popular German file format). Raster graphic files are .GEM metafile or *Calamus* file format.

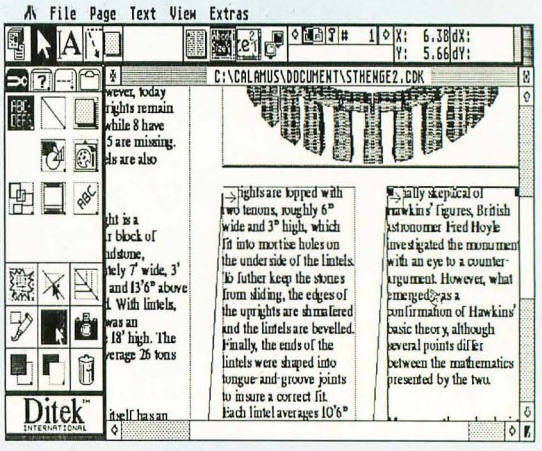
Lined up against a program like *Ventura*, measuring feature against feature, *Calamus* can comfortably hold its own. Some of the features may strike you as less than satisfactory however. For example, although *Calamus* can directly control scanners, two of those supported are not sold in North America (the Silver Reed and the Hawk CP-14) and the third (Panasonic FX-RS505), while available, has no ST interface. None of the best-selling scanners on this continent are supported. ISD accepts the scanner interface on faith, since they have only been able to test the Silver Reed—which wouldn't work with either *Calamus* or its own ST scanning software!

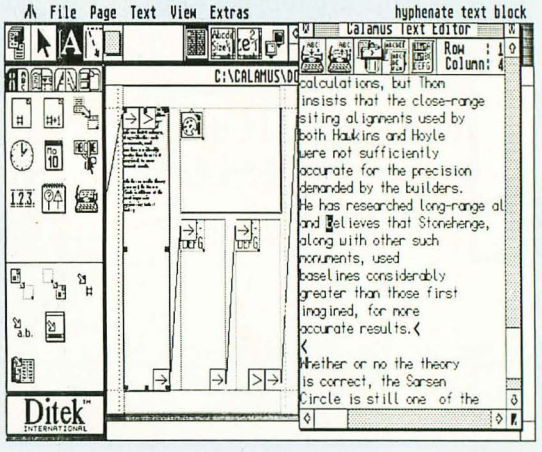
Other features fare somewhat better. Virtual copy is a godsend for designers who need to create and work in multiple-copy works such as business card templates. It essentially reflects any change to the original or any virtual copy to every other virtual copy.

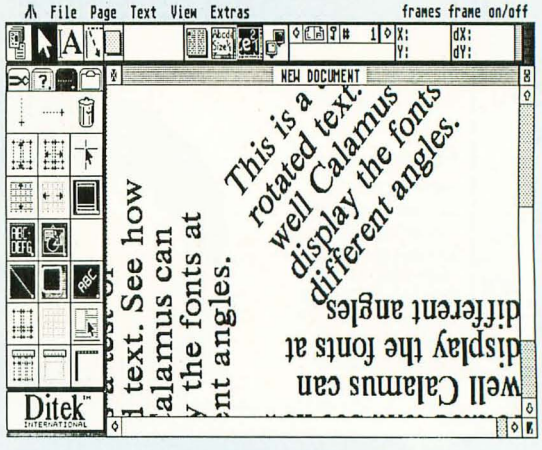
There is a macro feature in which text rulers, text style and text itself can be saved as a control-key macro and used in a similar manner to *Ventura*'s paragraph tags, to apply a style to a body of selected text quickly and easily. Unfortunately, the programmers neglected to include the simple shift-click means of highlighting a block of text as in many GEM applications.

The designers of *Calamus* made a special effort to overcome the restrictions created by the inept design of the Atari SLM804 laser printer. The printer was built without any memory or the possibility of adding any—a marketing device, no doubt, to push Mega ST sales. Laser printers require at least one megabyte of RAM to process a graphics page, and because of the printer's poor design,









owners of 1040 STs are unable to use it except as an overpriced text-only printer.

Not so with *Calamus*. On a 1040 ST, you direct the output to disk. Then you shut off the machine and reboot, without any desk accessories or other programs loaded, with only a small program called DISK__PRT in an auto folder. This program takes the output files and uses all of the available memory in your 1040 to process the print for the SLM804. Not a very elegant method, but certainly DMC should be applauded for coming up with a solution.

Printer drivers provide adequate, if not extensive, support for 8-, 9- and 14-pin dot-matrix printers, as well as the HP LaserJet Plus, the most commonly used laser printer in the U.S. However, there is an enormous range of printers not supported—the enormously popular Epson LQ series for example—and one can only hope that the programmers will provide drivers for them in the very near future.

There is a large hyphenation dictionary for text, but ISD was unable to name the source. The hyphenation appears adequate, but without knowing what authority created the word breaks, I'll only use it with a known dictionary (Merriam Webster's, Random House, or Oxford University Press) beside me.

Working with *Calamus* can prove to be initially bewildering. It uses an icon-based interface, along with the standard GEM menus (and many enhanced-GEM dialog and file selector boxes). Many icons lead to one or more levels of additional icon pads. At the top of the screen, a brief description of the purpose of an icon appears, so you don't need to memorize them all. These messages, by the way, can be changed by any user by editing the text files provided.

After a few hours, the interface becomes second nature and proves quite easy to use, although *Calamus* is overly mouse-dependant and lacks keyboard alternatives for many of its functions. You can apply some of the commands to the function keys by editing the various ASCII help and message files in the system, but it seems that this forces the user to do what the programmer should have done originally.

Frames can be set to exact X,Y coordinates, which makes layout more precise. Text frames can be set to flow around other frames (text or graphic) for text runaround on either side of the frame. If you move a frame from a runaround, the runaround area remains intact until you re-select other frames for runaround. This permits you to then offset the first frame to create white space between frames. Frames can be inserted into or cut from a piping chain.

It's easy to simply list the features, but that doesn't give the reader the true picture of either what *Calamus* can do or what its weaknesses are. You have to work with it to fully appreciate the richness inside. It's hard not to compare *Calamus* with *Ventura*, but overall, I'd say it provides a better working environment than the latter. Certainly the display capabilities are far superior. If the weaknesses can be overcome, *Calamus* could become the most popular DTP program around, on any computer system.

Problems

Calamus is not without its flaws and bugs. It crashed on several occasions while I wrote this review, although I was not always able to duplicate these problems in subsequent tests on a slightly later version.

The flaws are another thing. DMC put an enormous effort into the development of the main features, but appears to have glossed over many small, user interface issues. Many dialog boxes, for example, are missing a simple "cancel" or "abandon" button. They also made several design choices which run counter to the standards set in the more advanced PC DTP market.

Undo should reverse the last action taken. As far as I can figure, Undo has no inherent function in *Calamus*. Without a working undo, it's far too easy to make fatal mistakes, such as accidentally deleting the wrong column, after spending hours formatting and editing.

The Delete key should delete a selected frame. It, like Undo and Help, has no function except in the text editor.

There is no "greeking"—using symbols to simulate text and speed up the display process. Even at unreadable sizes, *Calamus* attempts to properly display text as characters. This is unnecessary. You can, at least, turn off the display of any type of frame and just leave the frame boundaries on screen.

Calamus continues the curious and annoying practice in the ST world of forcing the user to create each frame for multi-column text individually, rather than being able to designate a single text frame as multi-column as you can in *Ventura*. One must then create "piping" channels to direct text flow. While not overly difficult, it is certainly cumbersome and primitive compared to *Ventura*'s elegant process. It also prevents the user from selecting text across multiple pages or frames.

There is no way to anchor graphic frames to particular text, although you can group frames together. There are no automatically generated caption frames for graphics. They must be manually created, as individual frames, then grouped with the graphic.

Frames can't be printed. You have to create a raster frame around the existing one, another minor but time-consuming annoyance.

The "index" function is nothing of the sort, and the results produced are so unsatisfactory that DMC and ISD should issue abject apologies for misleading users as to what they're getting.

The manual lacks an index, which makes trying to find anything in the book simply horrendous. This is an inexcusable lack in any program that has even minimal pretenses to professional use, let alone something of the calibre of *Calamus*. The manual is graphically well produced: It was done in *Calamus* itself and shows its capabilities, but neither the index feature nor the hyphenation dictionary were used, a telling comment on these functions.

The manual is sometimes helpful—in the discussion of fonts, and raster versus vector graphics, for example—but in most other areas it fails to provide the necessary information in sufficient depth. It lacks a tutorial, glossary, technical appendices on file formats, error message explanations, and the descriptions and explanations are often simply not detailed enough for the user to understand without patient experimentation. There is no online help system to supplement the manual. Indeed, the Help key performs no function at all in *Calamus*. Finally, the grammar is often bad (owing, I suspect, to a faulty translation from the German), and there are typos and spelling mistakes in the book.

I was able to spend several hours with Nathan Potechin of ISD, going over the program, which considerably heightened my respect for *Calamus*. Most users are not afforded this luxury and must depend on the inadequate manual for their training. ISD is well-advised to scrap the current documentation and rewrite everything from scratch.

Faults notwithstanding, *Calamus* is an amazing program, and the features and functions outweigh the problems. Development is ongoing and changes are being made to the code by DMC in Germany. Unfortunately, we in North America don't benefit immediately from its efforts. It is also working on a *Calamus Plus* package, with an improved text editor, vector and raster graphics/paint packages, proper indexing, and more.

It's unfortunate that ISD/Ditek chose to enclose *Calamus* in a plain white box without some sort of explanation on the wrapper of the features and functions supported by the program inside. The buyer simply will not be able to fathom the power of the program from the box. That's a poor marketing decision; *Calamus*'s light is too bright to be hidden under a bushel! ■

Ian Chadwick is a Toronto-based freelance technical writer, specializing in desktop publishing and word processing.

Arkanoid

Taito Software
11715 North Creek Pkwy. So., Suite 110
Bothell, WA 98011
(604) 984-3040
$34.95, color only

Reviewed by Frank Eva

It's been a long time since ST owners were treated to a *true* coin-op conversion. Does *Arkanoid* fit that bill?

Inside the *Arkanoid* package, you will find a comic-book-style mission briefing. *Arkanoid*, like most other pure arcade games, certainly does not need historical/philosophical justification for addictive play. It's just thrown in for laughs. It seems that you are in charge of the spaceship *Arkanoid* that has blasted away from Earth with the last survivors of civilization. The ship encounters a grid wall of dense, undulating energy. Suddenly, this energy materializes into a living force-field creature that blocks your path. It is impossible to turn back. Your only hope is to take a handful of your crew and elude the grid monster in the Vaus escape module. That's about it! What this has to do with *Arkanoid*, the game, I'll never know!

What I do know is this: *Arkanoid* is exactly what they say it is: one of the most addictive video games ever. The first time I saw *Arkanoid* in my local video-game emporium, I was not impressed. "Who needs another *Breakout* clone?" said I. While *Arkanoid* resembles *Breakout*, it is surely not a clone. Yes, you must control a paddle, the "Vaus," maneuvering an energy ball in order to break down walls of bricks, but there are many more twists that make this game fun and highly addictive.

One of the variations is the bricks themselves. Silver bricks will require more than one hit to disintegrate. Gold bricks are indestructible. Then, there are the ever present Konerds, Pyradoks and Tri-Spheres that appear on the playfield. At first glance, these appear harmless. But they can cause you trouble if you don't watch out. These animated sprites move toward the Vaus in their own attack patterns. They can be destroyed very easily, by contact with the Vaus or by being hit with the energy ball. They can also change the course of the energy ball, when they are hit. This is where caution must be utilized. If you hit one of these baddies toward the bottom of the screen, you may have the ball returned where you cannot reach it.

And finally, there are the capsules that are randomly released when the bricks are destroyed. Each are labelled with a letter related to the function of the capsule. Catch one, and it imparts its unique power to the Vaus. "B" (break) opens a door at the lower right-hand corner of the play screen. Moving through the door advances the player to the next level. "C" (catch) turns the surface of the Vaus sticky, so that the

The ST GAME

energy ball can be caught in mid-flight. This enables a player to take a breath, while he decides where to fire the energy ball again.

"D" (divide) splits the energy ball into three clones. This is useful when it has gone behind a wall. It can be disastrous if it happens in front of a wall, because in the heat of trying to catch up, you can make the wrong decision and wind up losing all three. "E" expands the Vaus to twice normal size, which makes it easier to hit the energy ball. "L" arms your Vaus with twin lasers that are fired with either mouse button; a highly prized capsule, when faced with many silver blocks. "P" (power) awards the player with an extra Vaus. Lastly, "S" temporarily slows down the speed of the energy ball. Only one capsule's power can be retained. Once another is taken, the original power is lost. The real treat here is the decision that the player must make during the heat of combat. "To take a capsule, or to not take a capsule; that is the question!" Frequently, you will find the energy ball and a capsule in the same area of the screen. If you go for the capsule, you could lose the energy ball, and vice versa. Probably, you will try for both, and come up empty-handed, while losing a Vaus to boot.

Several home versions of *Arkanoid* are accompanied by a custom paddle controller, that mimics a track ball. The ST already comes with a very decent substitute, the mouse. It is especially nice when used in conjunction with a mouse pad. The movement of the Vaus is smooth and responsive.

Arkanoid presents all of the 33 screens of the original arcade game. No codes are provided, however, that will allow the player to enter a higher level later on. You must always begin at screen No. 1.

Now, to answer the question I proposed earlier. Yes! *Arkanoid* is an excellent conversion of the coin-op original. The graphics are crisp and colorful. The animation is smooth and fast. The sound track is raucous and provides a real arcade atmosphere. The sound effects during the game are very realistic.

On the downside, the high-scores cannot be saved to disk, but since the disk is copy protected, this missing feature actually protects the disk from damage. I discovered a bug in the high score routine, anyway. After playing for awhile, I amassed a score of KK4000, and this score replaced the high score of 50,000. After that, I consistently scored more than 4,000, but did not get a chance to enter my initials on the high-score screen.

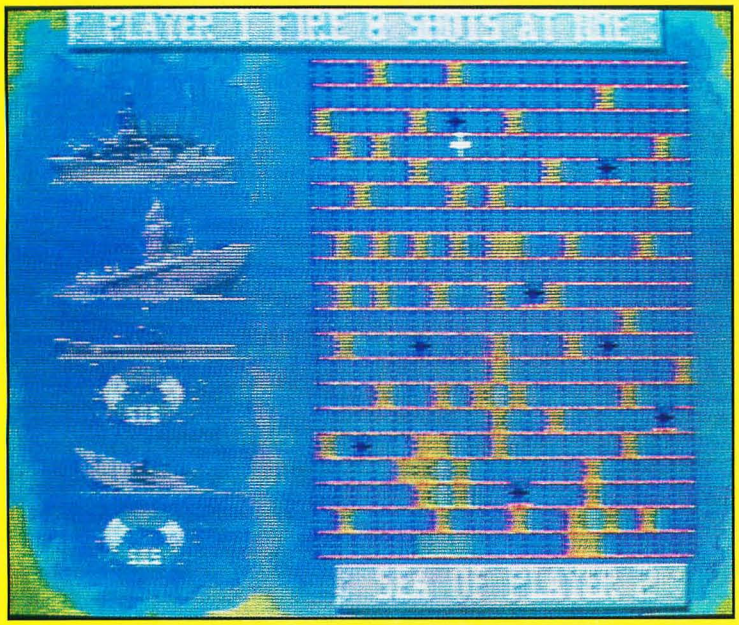
But this small bug and the lack of an on-disk high-score feature are minor quibbles. All things considered, Taito gets a B+ for its efforts!

■ **Recommendation: Buy it.**

SHELF

Battleship

Epyx
600 Galveston Drive
P.O. Box 8020
Redwood City, CA 94063
(415) 366-0606
$29.95, color only



Reviewed by Steve Panak

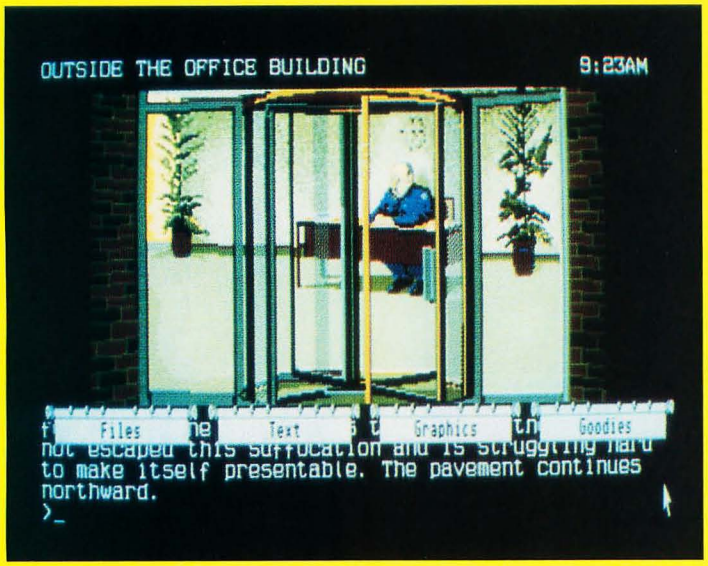
I go into most products with an open mind, trying not to be unduly influenced by my first impressions, attempting to evaluate the program on its own merits, disregarding any market hype. But going into *Battleship*, I expected the worse. And once I had this new computer version, sanctioned by Milton Bradley, in my hot little hands, my first thought was *Why?* While easily converted to computer, the game still seems best suited for play by two human players, and I rarely find thrusting a computer interface into such an equation resolves it satisfactorily. Boy, was I wrong.

I learned this immediately after booting up. Sticking closely to the formula of the perennial favorite, I first placed my ships on the battle grid, while my opponent did the same. Then came my first surprise. Rather than allowing me to fire one shot at my adversary, I got to pump out 16. Using the mouse I floated the targeting cross hair over my opponent's ocean, laying down a wide pattern I hoped would hit one of his ships, allowing me to home in for the kill on my next turn.

(Continued on page 92)

Corruption

Rainbird Software
3885 Bohannon Drive
Menlo Park, CA 94025
(414) 322-0412
$44.95, color or monochrome



Reviewed by Frank Eva

Think of text adventures and Infocom no doubt comes to mind. (Infocom is now under the auspices of Mediagenic, distributors of Rainbird products.) Now, who comes to mind when you consider illustrated text adventures? Chances are Magnetic Scrolls and Rainbird Software.

Magnetic Scrolls has produced *The Pawn*, *Guild of Thieves* and *Jinxter*. This latest effort is probably its most "adult" offering. There are no mythical lands, sorcerers or elves here. *Corruption* is a true-to-life adventure.

Imagine this: Your name is Derek Rogers, and you live in London. You are an ambitious yuppie on your way up the business ladder. Your ability at stock market manipulation has caught the eye of David Rogers (no relation), who makes you a full partner in his firm, gives you your own office, a high salary and your own BMW. You are at the top, but just as soon as you begin to celebrate your achievements, things start to unravel.

(Continued on page 92)

Gold of the Realm

Magnetic Images Co.
P.O. Box 17422
Phoenix,AZ 85011
(602) 265-7849
$39.95, color only

Reviewed by B.D. DeMunn

Are you ready for another endless trek through stone corridors or has dungeon burn-out rendered your loins ungirdable? Frankly, if Croc Dundee asked me for a date, I'd plead a headache. I've abandoned so many people in dungeons that I've been indicted for hero abuse. Nevertheless, here we go again.

The package illustration gently hints at what to expect: Young male in designer rags carrying candle. Key on floor. Bend in corridor. Stairs. I can hardly pop the shrink-wrap I'm yawning so violently.

As always, I scan the 24-page manual before I boot the disk. The story begins: "Nigel knew it was going to be an odd day when he awoke that morning." It ends: "Nigel said aloud, "I shan't ever be a prince, but with the Gold of the Realm, I would surely be a wealthy peasant indeed." In between, a stranger dies on Nigel's doorstep, but not before he tells him (*choke, gasp*!) about four castles, lots of gold and some evil stuff too. So greedy little Nigel, hurriedly buries

(Continued on page 93)

Jet

subLOGIC
713 Edgebrook Drive
Champaign, IL 61820
(217) 359-8482
$49.95, color or monochrome

Reviewed by Ron Schaefer, M.D.

SubLOGIC has recently released a realistic jet fighter combat simulation called *Jet* for the Atari ST. If you loved the movie *Top Gun* and have always fantasized about aerial combat beyond the speed of sound, then this is the program you have been waiting for.

Jet comes on one single-sided disk with a nicely done, 55-page paperbound manual that does a good job describing the somewhat complex controls, the rules, aircraft technical data and a brief history of fighter jets.

The program will auto-boot or can be started from the desktop. After a title screen you can choose from ten different playing modes: demonstration mode, F-16 Fighting Falcon dogfight, F-16 target strike, F-16 free flight, F-16 combined attack, F/A-18 Hornet dogfight, F/A-18 target strike, F/A-18 free flight, scenery disk load, and multiplayer dogfight. Next you select the difficulty, ranging from practice mode, where you can't crash and no enemy fires on you, to almost impossible, where missiles come

(Continued on page 93)

(Battleship Cont.)

After I had targeted my last shot, the grid was replaced by the image of my shots volleying toward my enemy's boats. As I watched from the deck of my gunship, one by one my shells lofted toward him, until finally a shot hit his aircraft carrier and crippled it. The buckled boat listed sickly to one side as it began to take on water. My magazine spent, the computer took its turn and missed me completely, but at this point I noticed one subtlety not in the original. Since he had suffered some damage in my barrage, he was allowed not 16 shots, but only 14. Near the end, as we searched for each other's small torpedo boats, we each had only four shots. This both increased realism and slowed play near the end, as every shot could be the last.

Technically, the computer proved an adequate opponent, and, once it scored a hit, it zeroed in on the wounded prey with vicious accuracy. Options allow you to select one- or two-player competition, as well as a tournament mode that allows for more than two players. For those who want to slow play, you can adjust the firing to a maximum of an entire magazine, or only up to four shots per round. Great graphics, realistic sound and a spartan, but complete manual round out the package.

So, although it's hard to believe, I'm going to have to recommend *Battleship*. While at first glance it seems like a cheap rip-off of a childhood favorite, it is instead a competent computerization of a classic board game.

■ **Recommendation: Buy it.**

(Corruption Cont.)

Included in the package is an audio cassette. The story contained therein (the cassette contains the program's original sound track, too, that double-sided drive owners can hear through their monitor's speaker, at the beginning of each game), coupled with suggestions on the box, paint a fairly grim picture of your first day at the new job. You have become the chief suspect in an insider trading scandal—and no one is rushing to your defense!

(For the uninitiated, insider trading is an illegal stock market activity where the broker uses privileged information to make investments for clients or himself, which results in huge profits.)

The ST version of *Corruption* features 80-column text and excellent painting quality illustrations that can be pulled down, nondestructively, over the text. You can, in fact, play without the use of the illustrations altogether, for they do not figure heavily in the actual game play. For example, in an early scene, you can plainly see a cleaning lady, but try as you may, the program keeps telling you that she just is not there. Obviously, at this point in the game, her part in the adventure is only atmospheric.

Pull-down menus also provide for some other nice extras, such as an item to instantly reveal all the exits for the current location. The item selected (highlighted) is activated by a left mouse click. To pull down the illustrations screen, you must point to the menu bar and hold down the right mouse button while pulling. Monochrome monitor owners will welcome the ability to size text or alter the display through techniques including stippling and dithering.

The documentation includes a quick start card that provides loading instructions, an adventure guide (quick reference card) detailing special functions, and a game play manual that explains methods of communicating with the game, as well as providing encrypted hints. Now, you might view the use of hints as a form of cheating, but believe me, you won't want to use these hints unless you're desperate. The smallest cipher consists of 14 two-letter codes, while most require 30 or more. Each cipher must be entered without a single error in order to receive a useful hint.

The player is allowed to define the ten function keys to frequently used phrases. This should save typing time, but since there is no way to save your definitions, this feature will only be valuable if you play for a few hours at a time.

The parser's vocabulary is not tremendous, but should suffice. The main thrust of the game is away from object manipulation and relies heavily on character interaction. There are 15 personalities that the player must learn to communicate with if he/she expects to complete the game. Each action adds one minute to the game clock. This means that certain developments are going to happen at specific times. You will have to be in the right place at the right time in order to witness the action. It also means that the player has a limited time before being hauled off to jail. Once there, the game is over. Maintaining your freedom is essential in order to untangle yourself from the long arms of the law that tighten their grip on you, the chief suspect.

As mentioned earlier, this is an adult adventure. This does not mean that the language or illustrations are of a questionable nature. It just means that this is not a game that the kiddies will readily comprehend. The themes are adult, and the game play moderate to difficult.

A problem that arises with software that comes in to this country from overseas is the cultural barrier. Certain expressions will leave the player somewhat puzzled. For example, in one scene you are told that the secretary has made her office "homely" with a vase of flowers on her desk and family portraits on the walls. Obviously, over in the U.K., "to make homely" means "to make it like home." On this side of the big water, we think of homely as unattractive. This cultural barrier is only a minor inconvenience however.

To be successful at *Corruption*, just remember that you are thrown into the world of cutthroat business, with all of its power plays, greed, deception and double dealing. You must become ruthless in order to survive.

Adventure gamers who are tired of slaying dragons will surely find *Corruption* an intriguing change of pace!

■ **Recommendation: For graphic adventure fans.**

(Gold of Realm Cont.)

the poor guy in the north 40, (without so much as an "Our Father") and boogies forth to find wealth, happiness and the medieval dream.

I'll let you decide whether you want to hook up to a compatible synthesizer or keyboard and share the music of *Gold of the Realm* with the next county. MIDI is still a sailor blouse to me. But the manual details the techniques involved, and I must admit, this is a new and unusual option. My ears were primed for some fancy tone painting, but the music is hardly memorable. In fact, when Nigel gets killed, as he often does, what do we hear? *Barf.* The Funeral March. Is there no other tune to die to? Anything by Ozzie O. would do.

Having three levels of difficulty is nice for a change: easy, medium and hard. "E" dumps you in one gray castle with 79 screens to explore; "M" has gray and green castles, 159 screens; and "H" adds two more castles for a total of 320 screens.

So you, (as Nigel), enter a castle and get "stoned." Use both joystick and mouse for movement and manipulating of objects. If you care to map, as the manual suggests, it's one helluva job, because each stairway leads to a different area of the level above or below. And, of course the lower level is dark, so find that candle.

A "spook" wafts by occasionally and rips off the object you're carrying, relocating it who knows where. It also slurps your strength bar, so munch apples for a healthful pickup. Pills and potions abound, each with a specific use that is not immediately apparent. You must find the scroll to get a hint. As you're allowed to carry only three objects at first, you have to intuit which three, and find a convenient room to stash the extras for future use.

Without so much as a spitball, there's no defense against unfriendly beings except the magical items you pick up and speedy Reeboks. In some outdoor scenes you have to pussyfoot past snipers with Uzis. The villains in the "hard" option actually chase you right out of the castle. Some corridors are blocked by "castle mist" (i.e., force field), which sucks yoiur strength like soda through a straw.

My Nigel came through one walking like Tim Conway's old man character. His little arms and legs were swinging like crazy while he was practically at a standstill. But humor wears off and irritation sets in.

And the keys! Green, red, blue, pink, gold, black, brown—but which color opens which door? Reminiscent of what old game? Was it *Montezuma's Revenge* that loaded you with red keys and blue keys? Anyway, there are lots of keys and lots of doors and never the twain shall match.

My ambivalence toward *Gold of the Realm* is understandable; I've been playing similar scenarios since *Wolfenstein* days—time and patience are running short. Younger adventurers will probably love it. I admit it has a certain challenge and charm about it, especially at the hard level, where a wizard causes a surprise transmogrification! Come to think of it, I responded very vocally to *Gold of the Realm* with muttered curses and groans and giggles. Something must have grabbed me besides my girdle.

"Deja vu—through and through. Not for moi; maybe vous."

(Largo) *Dah-dah-di-dah-dah-di-dah-di-dah-di-dahhhhhhh.*

■ **Recommendation: Get a demonstration before buying.**

(Jet Cont.)

falling like rain and your jet crashes easily.

In the F-16 dogfight mode, you start in your hanger, where you hit the throttle and taxi out onto the runway. You can toggle radar, a map, a second 3-D view, a targeting circle or a heads-up altitude indicator. One nice feature is that you can set up to ten custom window configurations that can later be recalled with a single keystroke. If you want a different perspective, you can look out of your jet in all directions, zooming in or out, or view your jet from the control tower or a spot plane. The graphics are 3-D solid raster, complete with shadows. To speed up the action you have the option of turning off the shadows or going to vector graphics.

Taxiing down the runway (using the mouse, the keyboard or your favorite joystick), you punch the throttle and lift off into the wild blue yonder—or gray yonder if you are using the program in monochrome. Remember to retract your landing gear to reduce air drag. Now spotting a squadron of enemy fighters on your radar, you bank your jet to meet them in combat. The MIG-21s are less maneuverable than your jet, but about the same speed. In the higher difficulty levels, you will encounter more MIG-23s, that are actually faster than you are.

Your first form of attack is your missiles. You have two types: AIM-9 Side Winders, medium range (5 miles) and heat-seeking; and AIM-7 Sparrows, longer range (25 miles) and radar homing. Maneuvering your jet into position, you line up a MIG-23 in your sights. When the MIG gets into range, a targeting computer shows when a missile hit is likely, and you fire. If you want, you can now turn on the camera in the tip of the missile and follow it kamikaze style.

A beeping sound alerts you to the presence of incoming enemy missiles—time for some evasive maneuvering. Jerking back on the joystick, you shoot straight up, spinning wildly. You pull nine Gs and black out momentarily, but the manuever works; your radar shows that the missiles have missed. Turning back into battle, you launch two more deadly Side Winder missiles reducing the enemy squadron to only two planes. You and the remaining MIGs circle each other in tight circles, and you take a couple missile hits, each one knocking your plane briefly into an uncontrolled spin. Blasting away with your machine guns, one more MIG bites the dust and disappears in a cloud of smoke. But from behind, that last MIG-23 has hit you. Your F-16 is critically damaged, sirens are sounding and lights flashing. Your only choice is to eject.

After you eject, you change your perspective to the tower, zoom in and watch as your parachute opens, and you float gently to the ground. You still have two more jets to complete this scenario, and for each 10,000 points, you will be awarded another jet. At the end of the scenario, you will be shown your points and what medals you have won, such as a Silver Star, Air Medal or Purple Heart.

In the F-16 target strike mode, your enemy consists of ground installations that you can detect on your radar. Your mission is to destroy them using AGM-65s Maverick optically guided missiles and MK-82 Smart Bombs. However, this is no duck shoot; as you approach these ground targets, you are bombarded with antiaircraft missiles which you must avoid. You can land back at your airport to repair any damage or to refuel and rearm.

The F/A-18 dogfight mode is very similar to the F-16 one, except that you launch from the deck of an aircraft carrier. The mission once again is to destroy the enemy fighters. There is added difficulty since now to refuel, rearm or repair, you must land on the carrier deck—no easy task. In the F/A-18 target strike, your mission is to sink a flotilla of enemy ships, again using your smart bombs and avoiding being shot down by the enemy ship-based antiaircraft missiles.

The F-16 free flight and F/A-18 free flight is for practicing acrobatics and landings, especially those carrier landings. You can also explore a scenery disk that you have loaded with the Scenery Disk Load option. These scenery disks are the same that subLOGIC uses for its *Flight Simulator II* program, including San Francisco, Dallas, Miami, Japan and Western Europe.

The final mode is Multi-Player Dogfight. This can be done in two ways: over the telephone at 300, 1200 or 2400 baud or by connecting two Atari ST's together with a null modem cable. You can order a null modem cable from subLOGIC for $4.95 plus $2 for shipping or, following the instructions, make your own.

In summary, I feel that *Jet*'s controls are difficult to master and should be improved. Even a small movement can send you wildly out of control. The graphics are good but jerky, especially when compared to *Carrier Command* or *Starglider II*. I did not enjoy the target strike missions nearly as much as the dogfights. The multiple player mode is nice, and I look forward to trying it over the phone lines.

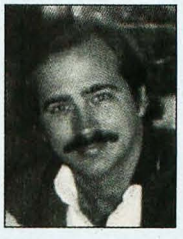
The game play in *Jet* is superior to some other jet fighter simulations I have played, including *Sky Fox*, *Harrier Strike Mission*, *F-15 Strike Eagle* and the newly released *Sky Chase*. I am anxiously awaiting the Atari ST release of the arcade hit *Afterburner*, and I have also heard that the author of *Starglider II* is working on a jet fighter simulation. But that is then and this is now, and it's back to *Jet* to see if I have the right stuff.

■ **Recommendation: Get a demonstration before buying.**

Frank Eva is an auditor by profession, but has been involved in the computer industry ever since his purchase of an Atari 400 many years ago. He has dabbled in programming and has had several text adventures published. His software reviews have appeared in several magazines.

Steve Panak has written more game reviews for ST-LOG and ANALOG than anyone on the face of the earth. He lives in Ohio where he plays games on his ST and, with the time remaining, practices law.

Betty D. DeMunn bops around Buffalo, New York, in her Yugo, chomping chicken wings and beef on weck. Evenings are spent killing and being killed. She used to crochet afghans.



Ron Schaefer, M.D., lives in Hawaii, where he specializes in internal medicine. He has published research on 3-D molecular modeling of proteins and DNA, as well as numerous articles on the Atari ST.

ScanArt, DrawArt Professional

Migraph Inc.
720 S. 333rd St. Suite 201
Federal Way, WA 98003
(206) 838-4677
ScanArt, $49.95
DrawArt Professional, $69.95

Reviewed by Maurice Molyneaux

If there is one computer-related term you can't seem to get away from these days it has to be "desktop publishing" (DTP). Those two innocuous-looking words have transformed much of the personal computer industry, changing machines that were once mostly used for keeping inventories and calculating the bottom line into publishing houses in miniature. The programs responsible for most of this revolution are the software equivalent of a typesetting machine and a layout and paste-up department. Your printer acts as a printing press.

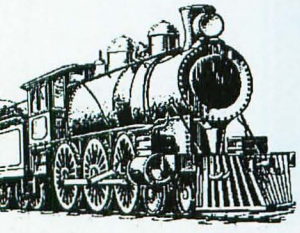
Simple, yes, but most DTP software stops there, providing only limited tools for the addition of elements besides text. You can sometimes draw simple graphics from within the program or import bit-images from another program; but the problem is that most people are not artists and not up to the task of creating polished-looking images to import into their documents. In real printing shops they have books and files of clip art: Stock images for all manner of needs.

Migraph, the people who brought us the excellent *Easy Draw*, have attempted to fill this need with two packages of "digital" clip art. The *ScanArt* and *DrawArt Professional* packages each consist of two double-sided disks containing many, many clip-art images for use in appropriate desktop publishing and/or drawing software. Each product is also slightly different, as you might guess from their names.

ScanArt is, as the name suggests, a package of scanned images. For those new to the subject, a scanned image is one in which a drawing or photograph is "read" by a scanning device, and its likeness is recreated in a computer-usable form. Thus, these images often look very much like "real" artwork when printed out on a decent printer. Proper use of such images can make documents more attractive, and, since the images are scanned and don't really look "computerized," they tend to make them look more professional as well.

The *ScanArt* disk contains over 100 scanned images, which are categorized and segregated in folders under the subjects of animals, bears (cartoon), food, holiday, humor, mortise, office, performance, santa, school, sports and transportation. The images themselves range from cartoony to realistic and cover a broad range of styles. It's much like having an actual clip-art book at your disposal.

And the images are quite good and generally print out very nicely, even on a lowly 9-pin printer like my trusty old Star SG-10. They are also in varying sizes (number of dots scanned), which, although they can be scaled up or down, all tend to have a bottom-limit to how small you can shrink them before the details begin to drop out and/or get muddled. This is not a fault of the supplied artwork, but a practical limitation of the maximum resolution of your printer. Also, generally you'd never scale one of these images up so large as to cause the



dots making up the image to become too obvious.

Each image is stored as a standard compressed .IMG (image) file and can be loaded into any application that can read these, such as *Easy Draw* (you must have the Supercharged version!) and *TimeWorks Desktop Publisher ST*.

DrawArt Professional is a package of drawn images, not scanned ones. These pictures are stored as standard .GEM graphic files and are usable by any program that can read these (any version of *Easy Draw* will work). Since the .GEM file format stores not bit-images, but the actual data for the manner in which the pictures are drawn, they can be rescaled, changed and manipulated almost endlessly (provided the program you're using has these editing tools).

Surprisingly, despite the fact that the over 150 images provided on this disk were drawn using a computer graphics program, most of the images are of high quality, many comparable to the scanned images on the *ScanArt* disk. The key difference is, of course, that as editable .GEM files these are much easier to customize than scanned images. For example, after loading the US__STATE.GEM file into *Easy Draw* you could break the group up into its individual components, grab and enlarge a specific state, and even make a dark copy of it "underneath" so it seems to have a shadow as it "floats" above the rest of the country!

The categories of art on the *DrawArt* disk are animals, arrows (three types), art tools, autos, borders (page), buildings, clip art (miscellaneous), computer, food, lights, maps, music, office, photo, sports, stars (graphic symbols, not movie stars!) and transportation. As with the *ScanArt* disk, the printout quality of these pictures is top-notch.

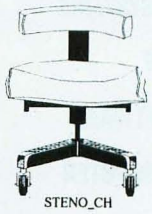
You may have noticed some overlap on topics between packages. It's true that both have clip-art topics of office, animals, etc. However, surprisingly, no images from either package are identical—or even very similar! There is no noticeable repetition.

Another added bonus is a measure of portability. Both products come on disks that can be read on both PCs and STs, so, if you use a program like *GEM Draw Plus* (*DrawArt* only), *GEM Desktop Publisher* or *Ventura Publisher* on a PC (either package), you can use the clip art there as well.

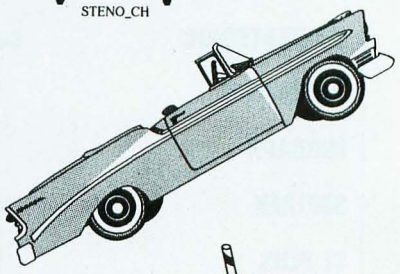
No review of these programs would be complete without a mention of their documentation. Both products come with documentation that describes not only what the images are for, but gives practical advice on use and limitations on use. Furthermore, there are printouts of every image included, so you don't have to load each one to see what you have. *DrawArt*'s documentation is bound up more in a "manual" form, while *ScanArt*'s is a series of stapled full-size sheets of paper. Each is fine, with text brief and mercifully to the point.

Now, some of you may be wondering which of these disks would be your best buy. I can't honestly tell you. Both are excellent collections with very useful selections of clip art. Sure, some of the scanned images are prettier and more detailed than some of the drawn ones, but some of the bold graphics and symbols of the drawn images are more commonly used by most people. Both packages contain sets of images that I think would be useful to the regular desktop publisher on a day-to-day basis. Furthermore, both are great products and both are worth having. If you have a need for clip art, and a program that can use .GEM and .IMG files, buy them. ■

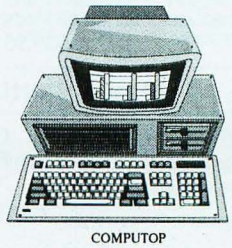
When not writing articles for ST-LOG *or otherwise working on computers, Maurice Molyneaux studies classic cel and modern computer animation, deadens his eardrums with overloud classical music, and further damages his already questionable sanity by listening to recordings of Monty Python, Tom Lehrer and Laurie Anderson. Otherwise he just makes a nuisance of himself. His DELPHI username is MAURICEM.*



STENO_CH



WORKDESK



COMPUTOP

INDEX TO ADVERTISERS

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

MAY 1989 · DISK LISTINGS

The ST-LOG #31 diskette contains 19 magazine files. They are listed below.

```
   FILENAME.EXT    FILE TYPE     COMMENTS
   ------------    ---------     --------

\ASMLINE\
   ASM_01   .PRG   RUN FILE      ASSEMBLY LINE
   ASM_01   .S     ASSEMBLY      SOURCE CODE

\LINEATTK\
   LNATTACK.PRG    RUN FILE      LINE ATTACK
   LNATTACK.S      ASSEMBLY      SOURCE CODE

\MONOGRAY\
   DEGAS_1 .GFA    GFA BASIC     MONO-GRAY, L1
   DEGAS_2 .GFA    GFA BASIC     MONO-GRAY, L2
   DEGAS_3 .GFA    GFA BASIC     MONO-GRAY, L3

\OUTLINE\
   OUTLINE .PRG    RUN FILE      OUTLINE PLUS
   OUTLINE .PAS    PASCAL        SOURCE CODE
   DRAW_MOD.PAS    PASCAL        SOURCE CODE
   IN_MOD  .PAS    PASCAL        SOURCE CODE
   MOD_CONS.PAS    PASCAL        SOURCE CODE
   MOD_TYPE.PAS    PASCAL        SOURCE CODE
   MOD_VAR .PAS    PASCAL        SOURCE CODE
   OUT_MOD .PAS    PASCAL        SOURCE CODE
   REC_MOD .PAS    PASCAL        SOURCE CODE
   RES_MOD .PAS    PASCAL        SOURCE CODE
   SUBR_MOD.PAS    PASCAL        SOURCE CODE

\PDPARADE\
   SAVELOAN.ACC    ACCESSORY     PD PARADE
```

DISK INSTRUCTIONS:
Only those files with .PRG, .TOS or .TTP extensions may be run from the GEM Desktop. Other programs may require additional software as shown below.

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

.EXT	DESCRIPTION
.BAS	Requires ST BASIC
.C	Requires C compiler
.PAS	Requires Pascal compiler
.S	Requires 68000 assembler
.GFA	Requires GFA BASIC 3.0 or GFABASRO.PRG

The Laws of Computing

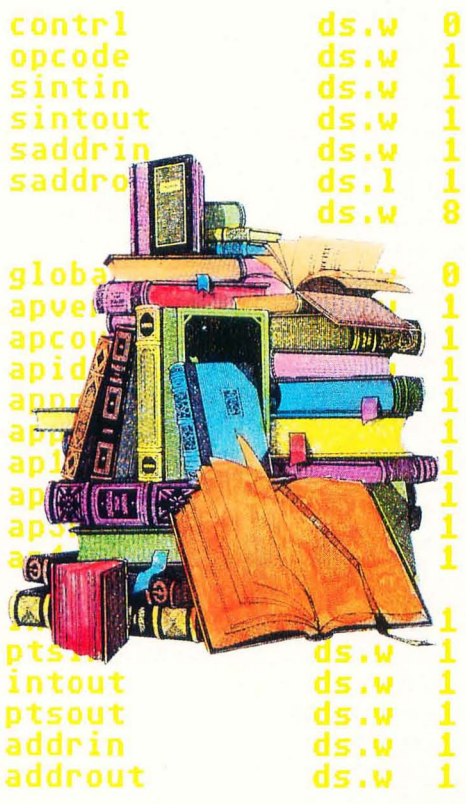
by Karl E. Wiegers

For 17 years, my fingers have caressed the keyboards of many computers. Ataris, Apples, IBMs, Compaqs, Control Datas and others have wrestled with me for control of the console. Sometimes I won, sometimes I lost. I win more now than I used to. In part, this is because I have become aware of certain immutable laws that govern the interactions of men and electronic machines.

I've formulated these principles as concisely as I can into Wiegers's Laws of Computing. No doubt there are others, but I haven't discovered them yet. Today I'd like to share these laws with you, in the hope that you, too, will win a bigger fraction of the console battles.

First law: *Almost nothing you can do with a computer is difficult.*

The novice laughs in disbelief when he reads this law. But it's true! Every computer program is just a sequence of short statements, most of which are easily understood by even a beginning programmer. And as a user, almost all commands you issue to the computer are simple, containing just a few words, and maybe punctuation, but you might not need the right parenthesis, and blanks, which may or may not be important, and quotation marks, which may have to be single or double. Sometimes you have to type in uppercase, but sometimes the computer doesn't care.



Now, none of these details results in a particularly difficult interaction. The hard part is remembering the precise format of each command. How do I erase a file: ERASE, DISCARD, PURGE, DELETE, KILL, FLUSH? Can I abbreviate the keyword? To how many letters? Or do I select a letter (D, maybe) from a menu? Or do I click on the file and drag it somewhere? The real goal is to avoid the computer responses that make our blood run cold: "Syntax Error," "No Such Command," "File Not Found."

You see, the difficult part is not issuing a command to the computer; it's remembering which command to use and how to use it—which leads to the next law.

Second law: *Almost nothing you can do with a computer is obvious.*

Once you've seen the second law, you're more likely to believe the first. The diversity of incompatible computing machinery and operating systems means that we must remember an inordinate amount of trivia to get our work done. I spend time these days on an Atari 1040ST, Atari 130XE, Apple IIe, Macintosh II, Compaq 286 and IBM 3090 (a huge main-

frame). This means I have to remember how to use six different operating systems, besides a ridiculous number of text editors.

The information I need for all these computers is not intuitively obvious. So in the absence of standard systems and online help, how am I supposed to keep all this stuff straight? With the aid of manuals, of course.

Third law: *The existence of manuals does not violate the second law.*

Naturally, this assumes that there *are* manuals, which is debatable for the Atari line. But the vast majority of manuals only pretend to help you remember the keystrokes you need to escape from your computer session alive. It's all in there somewhere (maybe), but I'll be hanged if I can find it when I need it.

I remember a time when the horizontal surfaces in one's office held things like pictures of the family. Now every square inch is filled with computer manuals, many still in the shrink wrap. Sometimes it's faster just to try all the commands you can think of, instead of digging out a manual and hunting for the answer.

But I have a theory about this. I think computer and software manufacturers deliberately write inadequate manuals, thereby creating a huge aftermarket for magazines, books, courses, keyboard templates, training videotapes and tutorial disks. And who writes all this stuff? Why, the wizards of course.

Fourth law: *If you are one week ahead of the other guy, you are a wizard.*

This is one of the social phenomena of the computer age. In more mature fields like woodworking, it takes years of experience to become an expert. But the rate of appearance of new computer information is so high that if you have just a little head start over the other guy, you look like a magician. Therefore you can be of great help to more primitive users who bought their computers the week after you did; you've carved a niche for yourself. The challenge for the wizards is to keep that one-week lead.

Fifth law: *If it is not in my computer and running, it does not exist.*

This law refers to the peculiar product category known as "vaporware." I'm sure you've seen ads or announcements for products that never quite seem to make it onto the dealer's shelf. Atari is notorious for producing both hard and soft vaporware. I always quote this law when someone tells me about the next release of *Superword* or how much easier life will be when *UltraMenu* arrives. *Ha*! It's safest to assume that something is totally imaginary until you have it in your hot, grubby hands.

Sixth law: *All programs are metastable.* "*Met-a-sta-ble, adj.* Designating a relatively unstable, transient, but significant state of a chemical or physical system. . ." (The American Heritage Dictionary.)

If something is metastable, a small perturbation can destroy it. A house of cards is metastable. Peace in the Middle East is metastable. And all you programmers know that if you tweak a program just a little, the whole computer might lock up. You won't even know what you did wrong lots of the time. See my ST-LOG series on software engineering for ways to violate this law.

Seventh law: *Artificial intelligence is no substitute for the real thing.*

People try to violate this law more than any other. There are times you need to rely on your own memory or good sense, rather than asking the computer to do the thinking for you. Users sometimes expect ridiculous sophistication from a program in the area of error checking, suggesting that they think the computer is smarter than they are. Faster, yes; smarter, no. The cerebrum is still the best piece of software we've got, folks.

Eighth law: *The fixing of one bug generates at least two new latent bugs.*

Again, you programmers will know what I'm talking about. The problem with the latent bugs is that they don't show up right away so you can squash 'em (that's what "latent" means). Of course, users are notoriously good at revealing latent bugs and loudly announcing them, which is why programmers in large companies always request (but never get) unlisted offices.

Ninth law: *Once you have trapped for all conceivable errors, the users become more creative.*

We software developers really do try to anticipate and handle all the things that might go wrong when a user runs our programs. But how seriously can I take someone who complains, if he leans his elbow on the keyboard when he starts my program, his disk drive explodes? Refer such users to the seventh law.

Tenth law: *The current backup copy of any file is always one generation too old.*

This, of course, assumes that you keep backup copies of important files. Naturally, you don't make them as often as you intend to. Hence, when the inevitable catastrophe takes place, the warm fuzzy feeling you get from knowing you have a backup disappears when you find the backup disk covered with cobwebs.

Eleventh law: *User manuals are consulted only after all possible keystroke combinations have failed.*

This is really a corollary (no, not a Toyota; look it up) to the Third Law. To be sure, some of the better manuals do contain the information you seek, but we seem to insist on trying everything before reading the instructions.

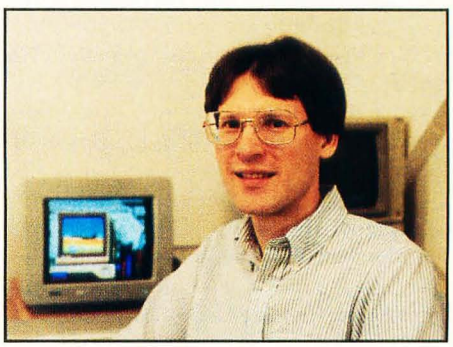
Twelfth law: *An undocumented feature is the moral equivalent of a bug.*

Every once in a while you'll accidentally encounter something extra in a program, a feature that isn't in the manual but is nice to have available. Why isn't it in the manual? Who knows. Although such discoveries are pleasant (if rare), I classify them along with bugs (which are unpleasant and all too common) from a philosophical perspective.

Thirteenth law: *Software version upgrades generally aren't.*

Perhaps this is mainly a phenomenon of the mainframe world, but I doubt it. All too often, a new version of an operating system or compiler results in your programs running more slowly and needing more memory. Sometimes existing programs won't even run any more. Sure, the software has more bells and whistles (which you'll probably never use), but it's rarely worth the aggravation. Bug fixes constitute a violation of this law, but remember the eighth law.

As I said, I'm sure there are other laws that haven't dropped into my lap yet, but I imagine they'll be along shortly. If you keep these fundamental principles in mind, maybe you'll have a bigger edge over the machine next time. ■



Karl E. Wiegers *pays heed not only to the aforementioned 13 laws, but also to the 14th law, which he forgot to mention: All regular* ST-LOG *contributors are popular with members of the opposite sex, are invited to lots of parties, don't have to eat quiche and are secure in the knowledge that their words are being read by the most intelligent readership in the industry.*

STPlus•STPlus•STPlus•STPlus

P.O. 1197, Berkeley, Ca. 94701 • add 3% credit card for hardware

Front line NEWS: GENLOCK for the ST, $400, preorder.

Spectre 128 (run Mac SE programs) $179 • PCDitto, DOS, & Drive $399

BUSINESS

DBMan 4.0	175.00
Datamanager	39.95
Superbase	104.95
Trimbase	69.95
Phasar 3.0	63.95
Zoomracks 2	84.95
Base 2	42.95
The Informer	69.95
Wordperfect	189.95
1st Word Plus	69.95
Word Up!	64.95
Best Accounting	279.95
Equal Plus	139.95
Inventory Mgr.	69.95
Rolobase Plus	63.95
Logistix Spread	104.95
Microlawyer	49.95
Payroll Master	69.95
Construction EST.	35.00
Microsoft Write	94.95
Datatrieve	35.00
STOneWrite	48.95
VIP GEM	104.95
DacEasy Payroll	48.00
DacEasy Acctg	52.00
WordWriter ST	56.00
SwiftCalc	39.95
EZ Calc by Royal	48.95
Analyze Spread	25.95
Final Word	99.95
PublishingPartner	140.00
T-works Publisher	89.95
EZData Base	48.95
Chart Pak	35.00
Compute Roots	27.95
Thunder NEW!	28.95
Habawriter 2	48.95
Text Pro	35.00
Becker Text	62.95
Expert Opinion AI	59.95
Time Link	35.00
Partner ST	48.95
Labelmaster Elite	35.00
ST Accounts	149.00
The Juggler 2.0	35.00
Max Pack	35.00
Stuff	27.95
Flash 1.5	21.00
Omni Res	27.95
Turbo ST(~blitter)	35.00
Signum *technical word processor*	249.95
SBT DLedger	195.00
SBT DPayables	195.00
SBT DInvoices	195.00
SBT DMenu	49.95
Neo Desk	27.95
Sales Pro	69.95
Mail Manager	39.95
Mighty Mail	35.00
First Word 1.6	14.00

GRAFHICS

Degas Elite	41.95
CAD 3D 2.0	63.95
Cyber Paint	49.00
Quantum 4096	27.95
Adv Art Studio	26.00
Spectrum 512	49.00
EzDraw&Superch	104.95
Canon Scanner	1040.0
GFA Artist 1000ci	55.95
Drafix 1	139.95
General Symbols	105.00
Elec, or Arch, Sym	105.00
Athena 2	69.95
Circuit Maker	55.95

CLIPART

Warriors(720)	14.95
Outdoor(720)	14.95
Buildings(720)	14.95
Victorian(720)	14.95
Etchings(720)	14.95
People (720k)	14.95
Politics(720k)	14.95
Religion(720)	14.95
Holidays(720)	14.95
Ad Art (720k)	14.95
Vehicles(720)	14.95
Boats (720k)	14.95
Planes (720k	14.95
All (9.2 meg)	99.95

GAMES

Gunship	35.00
Shadowgate	35.00
Uninvited	35.00
Mouse Quest	14.00
Slaygon	27.95
Barbarian	27.95
Obliterator	27.95
Gauntlet	35.00
Dark Castle	27.95
F-15 Strike Eagle	27.95
Star Trek-Rebel U	27.95
Questron II	35.00
Lock-On	27.95
Carrier Command	32.50

MUSIC

Passport	280.00
Master Tracks	104.95
MasterTracks Jr.	69.95
Midisoft Studio	
Hybrid Arts	
Smpte Track	499.95
Sync Track	299.95
EZ Track Plus	48.95
Midiscore	call
EZ Score Plus	104.95
DX-Android	139.95
CZ-Android	69.95
Gen-Patch	104.95
Dr. T's	
KCSequencer	199.95
KCS 1,6 w/PVG	289.95
MIDI rec studio	56.00
Copyist level 1	75.95
Copyist level 2	185.95
Copy3-Postscript	299.95

Remember: Every one thousandth purchaser gets a hundred dollars credit, and the ten thousandth purchaser will get a 10 meg Supra floppy. RULES: *Have your customer number or credit card ready. Purchase as often as you like. We're your computer supermarket and we're ready to fulfill your dreams but we're not an information service. We ship right away and its your part to know what you want. Don't be cheap, at these prices you can splurge!*

SPECIAL-SPECIAL-SPECIAL

MICROSOFT WRITE FOR ONLY $50

while supplies last.

HARDWARE

10 Meg Supra flppy	875.00
20 Meg w/clock	599.95
30 Meg Supra	749.95
60 meg Supra	1249.95
33 Meg Tulin	699.95
51 Meg Tulin	839.95
80 Meg BMS RRL	1249.95
20 Meg SH205	639.95
250 Meg	3250.0
10 Meg Floppy	849.95
AST PS LASER	3350.00
Canon Scanner	1040.00
IMG Scanner	90.00
ComputereyesMon	120.00
Supra 2400 modem	159.95
Atari SX212 modem	79.95

PROGRAMMING

GFA Basic 3.0	56.00
GFA Book	35.00
GFA Compiler	56.00
Mark Williams "C"	125.00
Laser "C"	159.95
Cambridge Lisp	139.95
RAID	27.95
Fast Editor	35.00
Alice Pascal	69.95
OSS Pascal	59.95
Fortran 77 GEM	139.95
BCPL	104.95
Modula 2 dev. kit	104.95
Assempro	48.95
Fast Basic	56.95
True Basic	69.95

EDUCATIONAL

Arakis Series	14.00
Unicorn Series	27-35
True Basic Stuff	69.95

Autumn 520 STFM Special

$599 Mono

$799 Color

Your Art Scanned $10 per page 75 to 300 dpi

The Best European Games

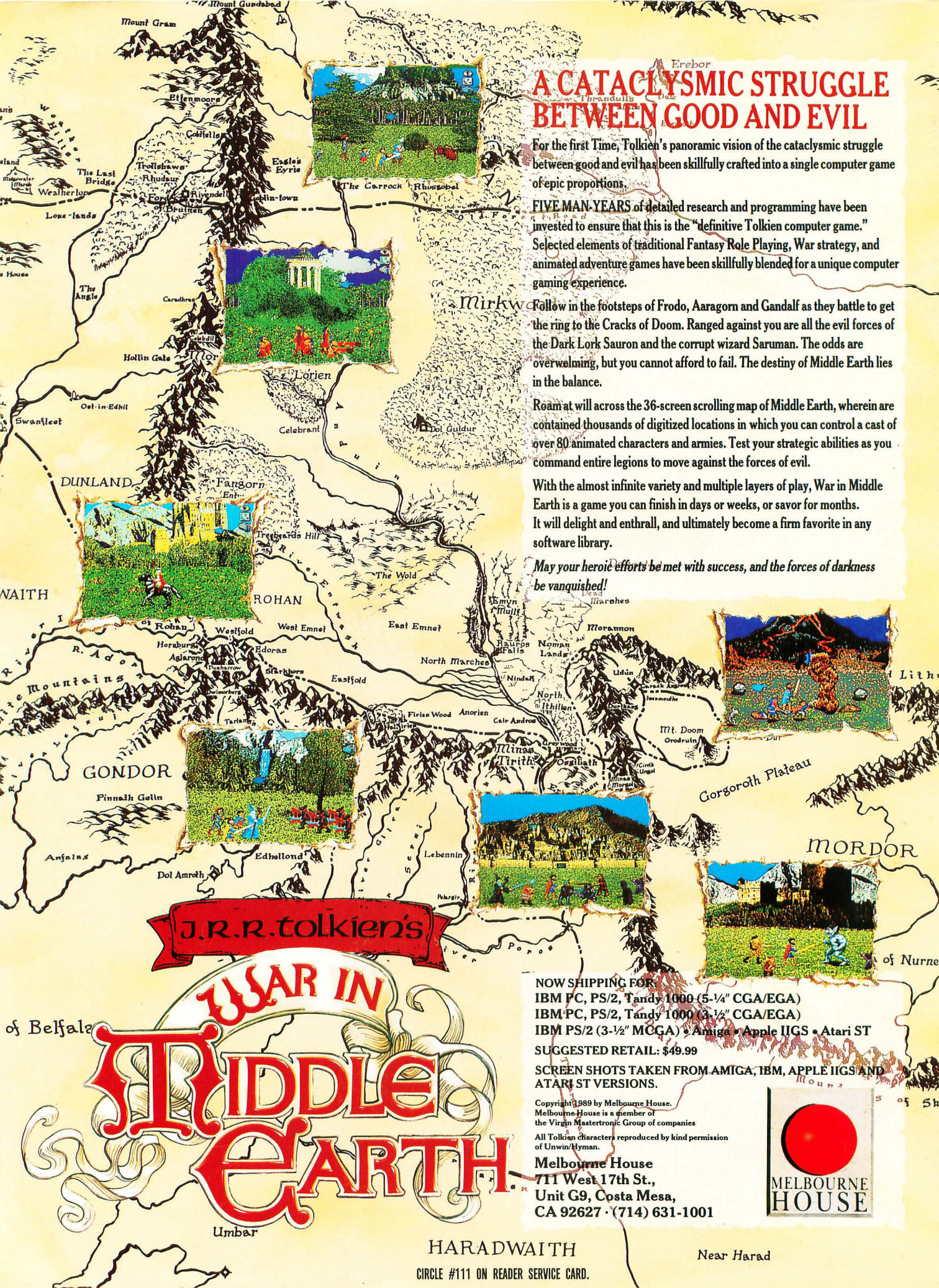
Tanglewood	27.95	I Ball (neat,fast)	19.95
Test Drive	35.00	RanaRama(d&d)	29.95
Chessmastr2000	32.95	Warlock'sQuest	29.95
Starglider II *NEW*	35.00	The Flintstones	29.95
Hunt for Red Oct	35.00	Trivial Pursuit	29.95
TyphoonThompson	27.95	The Enforcer	19.95
Aliants	24.95	Seconds Out	29.95
Alien Fire	35.00	Scruples(board)	29.95
Santa Paravia	21.00	Livingstone	29.95
Lurking Horror	27.95	Battle Ships	29.95
Star Fleet 1	39.95	Outrun(fast cars)	29.95
Empire	39.95	Crazy Cars	29.95
Liesure Suit Larry	27.95	Tetris (from USSR)	29.95
Gridiron	35.00	Screaming Wings	29.95
Dungeon Master	27.95	Spitfire	29.95
Flight Simulator	35.00	Blue War	29.95
Trailblazer	27.95	Star Quake	29.95
••••SPECIALS••••		Enduro Racer	29.95
Jewel of Darknss	19.95	BMX Simulator	29.95
Silicon Dreams	19.95	Arkanoids	29.95
Cardiac Arrest	48.95	Better Dead n Alien	29.95

Are you a gamer? How about joining our game of the month club? $10 gets you the hottest new title at an extra 5% off & you can return it for 75% credit. Plus you'll be eligible each purchase to win as #1000 or #10,000. Call us and be first to play the new ones. Start now with Typhoon Thomson for $25.95.

ONE NEW NUMBER **800-759-1110** Prices subject to change without notice.

We ship ANYWHERE! $4.00 min S&H. No 1040's or Megas mail order. Hand delivery only, List plus $100.

CIRCLE #110 ON READER SERVICE CARD.



A CATACLYSMIC STRUGGLE BETWEEN GOOD AND EVIL

For the first Time, Tolkien's panoramic vision of the cataclysmic struggle between good and evil has been skillfully crafted into a single computer game of epic proportions.

FIVE MAN-YEARS of detailed research and programming have been invested to ensure that this is the "definitive Tolkien computer game." Selected elements of traditional Fantasy Role Playing, War strategy, and animated adventure games have been skillfully blended for a unique computer gaming experience.

Follow in the footsteps of Frodo, Aaragorn and Gandalf as they battle to get the ring to the Cracks of Doom. Ranged against you are all the evil forces of the Dark Lork Sauron and the corrupt wizard Saruman. The odds are overwelming, but you cannot afford to fail. The destiny of Middle Earth lies in the balance.

Roam at will across the 36-screen scrolling map of Middle Earth, wherein are contained thousands of digitized locations in which you can control a cast of over 80 animated characters and armies. Test your strategic abilities as you command entire legions to move against the forces of evil.

With the almost infinite variety and multiple layers of play, War in Middle Earth is a game you can finish in days or weeks, or savor for months. It will delight and enthrall, and ultimately become a firm favorite in any software library.

May your heroic efforts be met with success, and the forces of darkness be vanquished!

J.R.R. tolkien's

WAR IN MIDDLE EARTH

NOW SHIPPING FOR:
IBM PC, PS/2, Tandy 1000 (5-¼" CGA/EGA)
IBM PC, PS/2, Tandy 1000 (3-½" CGA/EGA)
IBM PS/2 (3-½" MCGA) • Amiga • Apple IIGS • Atari ST

SUGGESTED RETAIL: $49.99

SCREEN SHOTS TAKEN FROM AMIGA, IBM, APPLE IIGS AND ATARI ST VERSIONS.

Melbourne House
711 West 17th St.,
Unit G9, Costa Mesa,
CA 92627 · (714) 631-1001

MELBOURNE HOUSE

CIRCLE #111 ON READER SERVICE CARD.