THE SECRETS OF ANIMATION

# ST LOG

THE ATARI ST MONTHLY M

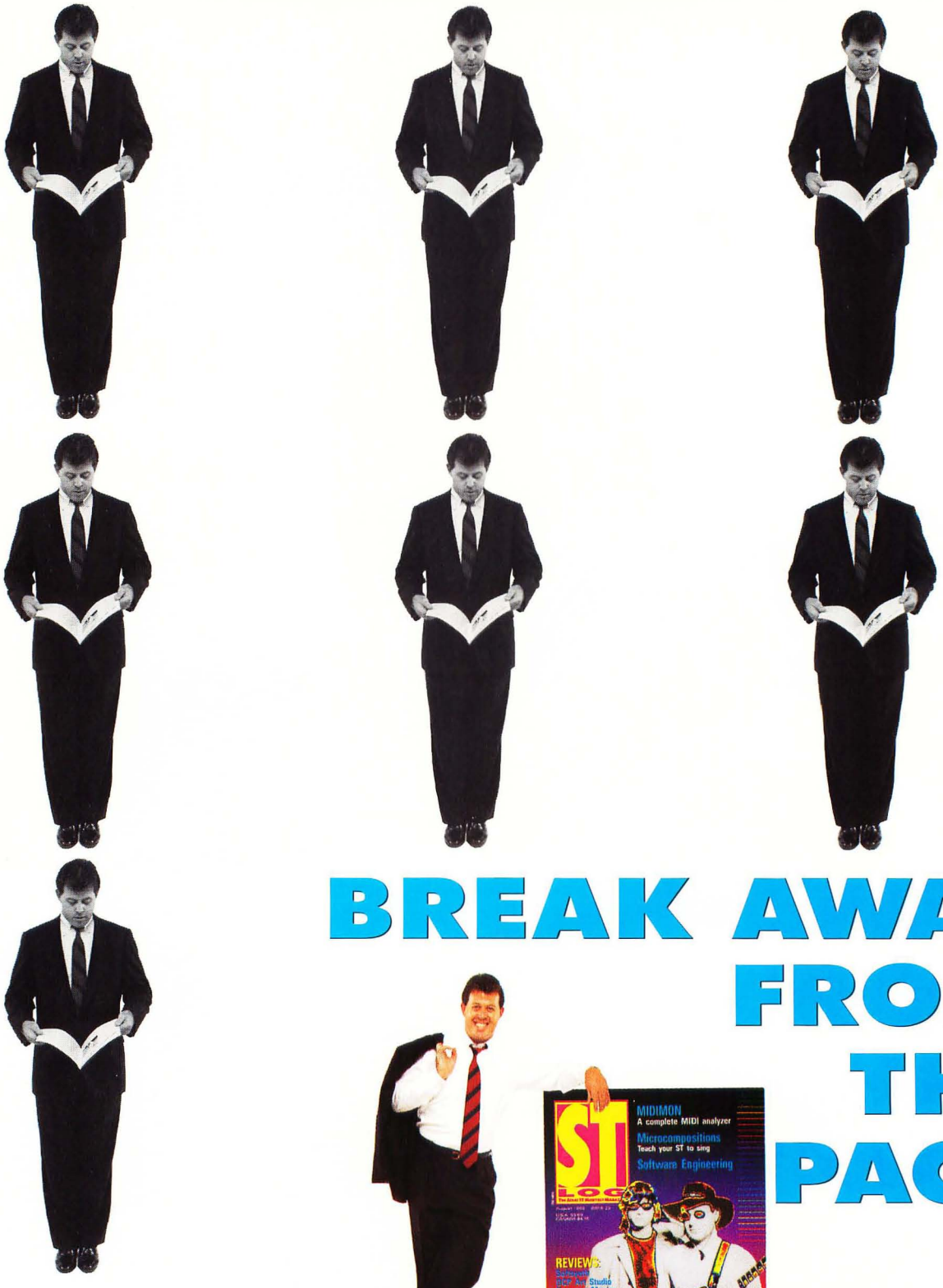
FDC 50076

APRIL 1989 ISSUE 30

U.S.A. $3.95
CANADA $4.95

## SOFTWARE ENGINEERING
## FONT ID EDITOR
## MULTIPAINT


74369 50076 1 04



# 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**

**$19 OFF THE COVER PRICE**

**12 Issues with Disk $79**

**NEW LOWER PRICE**

**FILL OUT COUPON ON PAGE 98 TODAY!**

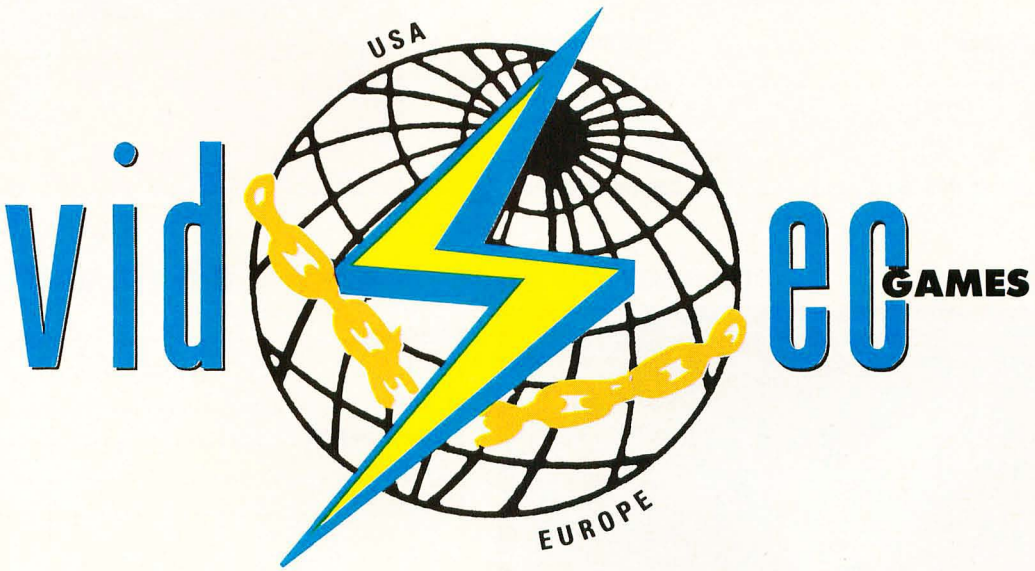
# EDITORIAL

BY CLAYTON WALNUM

As most of you know, there is a third entry in the ANALOG magazine group, VIDEOGAMES & COMPUTER ENTERTAINMENT. As the associate editor of this new magazine, I've had a chance to look at the video-game market from a new perspective, one that includes the entire market rather than just the Atari microcosm. Because of that, it amuses me whenever I hear anyone talking about Atari's video game image. It strikes me as ironic that Atari has all but lost that market.

Now I know that ST owners don't like to talk much about video games (although they buy more games than anything else), but this is important. Why? Try an experiment. Go down to the nearest toy store and line up the top three video-game systems. You'll be looking at the Nintendo Entertainment System, the Sega Master System and the Atari XE Game System. Now plug a cartridge into each and turn them on. Do you see how inferior the XEGS is to its two major competitors? Can you believe that, not considering the mail-in rebate Atari just started, the XEGS is actually priced higher than Nintendo's or Sega's machines? No wonder Atari enjoys less than 10% of the video-game market.

Okay, I'll get to the point. You all know about the 68000-based game machine that Atari is working on. What you may not know is that both Nintendo and Sega have new 16-bit machines almost ready to go—and that's not to mention another electronic marvel from Japan called the PC Engine that is due to be released soon in the United States. Not only does the PC Engine have full-resolution graphics, a huge color palette and dazzling animation, but it also has its own CD-ROM player. For those of you who don't know it, a single CD-ROM can hold something like 600 megabytes of data. Can you imagine the possibilities that opens up to the video-game world? (For a full-color preview of the PC Engine, see the April issue of VIDEOGAMES & COMPUTER ENTERTAINMENT.)

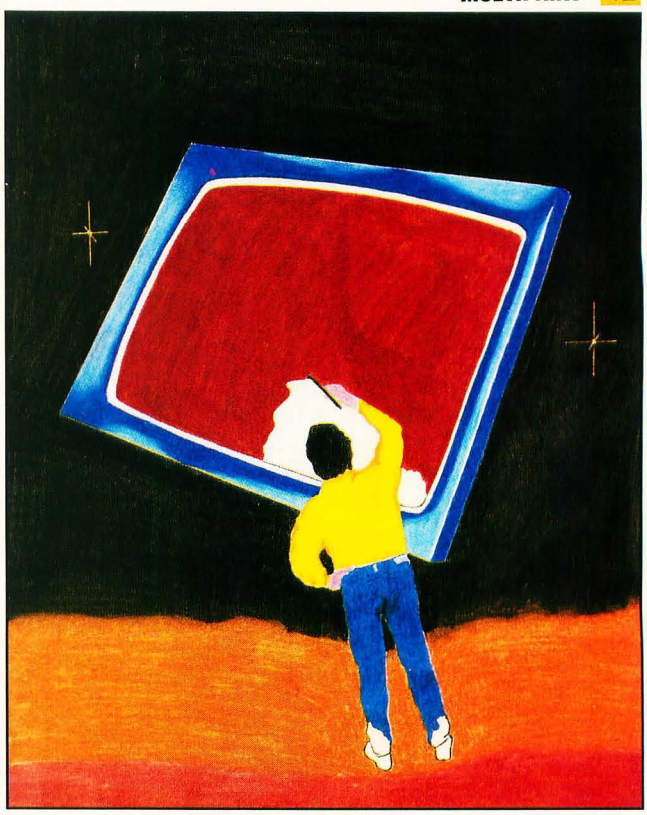



Okay, now I'll *really* get to the point. I know how everyone hates to see Atari associated with video games. But the fact is that Atari gets a lot of its income from those "toys," and I'm very much afraid that if we're to have a healthy Atari, it must be an Atari that sells video games. But one thing is certain: If Atari doesn't release their 16-bit game system soon, and if they don't immediately support it with lots of *new* and dazzling games, they are going to lose even more ground to Nintendo, Sega and that new whiz kid, the PC Engine.

Atari's current video-game products are already yesterday's news. They'd better start thinking about tomorrow.

# IN THIS ISSUE

MULTIPAINT 42



## FEATURES



## REVIEWS



THE ST GAMESHELF 84


## 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.

APRIL 1989
ISSUE 30

# 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**

**COVER MODEL:**
**SCOTT LANE**

**COVER ILLUSTRATION:**
**ALAN HUNTER**

**INTERIOR PHOTOGRAPHY:**
**JOSEPH DRIVAS**
**STEVEN HUNT**

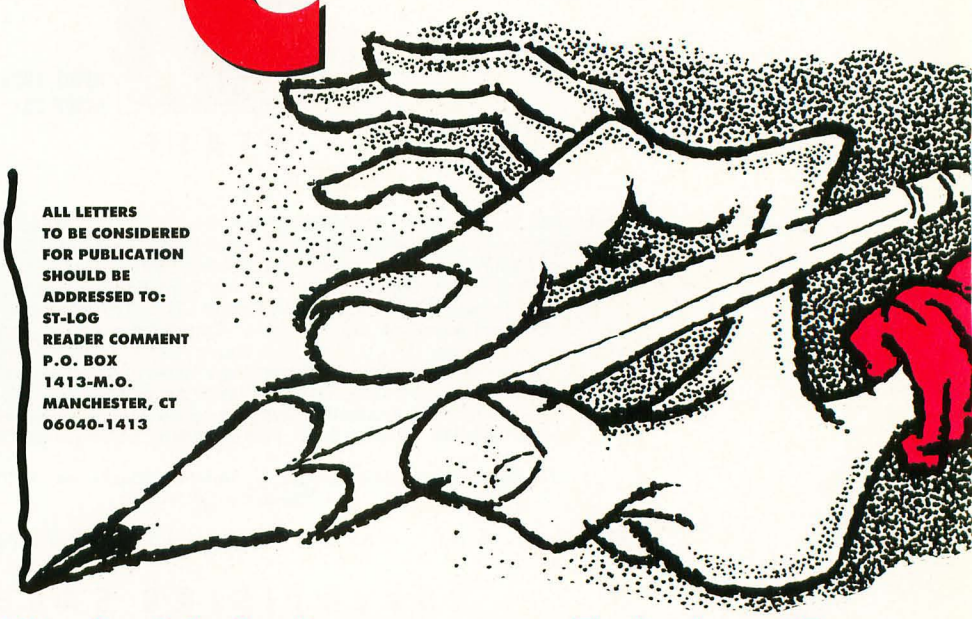
# SUBSCRIPTIONS

**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.

# READER COMMENT



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

## The ultimate language

In the *Ian's Quest* in the December '88 issue of ST-LOG, Ian stated that GFA BASIC was the best language available for the ST. I'm looking for a programming language more powerful than ST BASIC: one that I won't grow out of. In other words, I want a language with which I could eventually program a game equivalent to *Dungeon Master* or an emulator that rivals *pc-ditto*. Does GFA BASIC meet these standards? Or would I have to put in a bunch of machine language subroutines? If not GFA BASIC, what language would you suggest?

—Tim Hutchinson
Vashon, WA

***There is no doubt that GFA BASIC is currently the most popular BASIC language on the ST. However, when you state that you want a language that you won't grow out of, you're opening up a whole new set of questions. How fast you will grow out of a language depends to a great extent upon how far you plan to progress in your programming experiments. Since you mentioned both* Dungeon Master *and* pc-ditto *as examples of your goals, I would have to say that there is only one language that you are guaranteed not to grow out of: assembly langauge. GFA BASIC is capable of producing some very sophisticated programs. But GFA BASIC is, after all, a compiled language (in fact, GFA BASIC 3.0, the newest version of the language, is only, as of this writing, an interpreted language; the compiler hasn't been released yet), and that means that it will suffer, to at least some extent, in the speed and efficiency departments. When you're talking about writing something like an emulator, you really have no choice but to use assembly language. I can't say that it would be impossible to write a program like* Dungeon Master *in GFA BASIC, but it would be at least extremely difficult and would almost certainly necessitate some machine language subroutines.***

***When Ian said that GFA BASIC was the best language for the ST, I'm sure what he meant to say was that GFA BASIC was the best* high-level *language for the ST. The fact is that assembly language is the only language that will allow you to do anything that the hardware is capable of. Machine language (and by extension, assembly language) is the hardware's "native tongue."***

## Ultra-Graph in the classroom

You are to be complimented on the excellent article (and accompanying disk), "Ultra-Graph" by Blake Arnold and Phil Mast, in the November '88 issue. As a community college math teacher, I find it very useful in my advanced math classes.

Each year our campus has an open house, and I will use *Ultra-Graph* as a demo on a 25-inch screen. It should be a major attraction! ST machines are well suited for this kind of application; unfortunately, there is relatively little software of this type available.

Please continue to publish articles in this area. The needs of me and my students are: matrix operations, solving systems of equations, the Simplex Method and programs that simulate probability experiments. I know these exist on other machines, so it is possible for the STs.

—Hal Anderson
Santa Rosa, CA

***Thanks for the nice words on* Ultra-Graph. *We're sure that both Blake and Phil will be thrilled to learn that their program is being used in college classrooms. There is little doubt that* Ultra-Graph *is one of the best graphing utilities anywhere, including anything in the commercial market.***

***As for your other math program needs, we would be more than happy to publish other programs of this type. Any of you ST programmers who happen to be math experts should take up Mr. Anderson's challenge and see what you can create.* ST-LOG *would be interested in seeing your work. Just remember: the ST is a graphics-based machine, and programs should utilize those graphics abilities to the greatest extent.***

## The business ST

I recently read the article "Getting Down to Business with Atari" in the September '88 issue of ST-LOG and felt compelled to write concerning some business activities that I handle using my ST.

I am an assigner for the International Association of Approved Basketball Officials (IAABO), Board #11 in Wilmington, Delaware. I am responsible for assigning pairs of referees for close to 3,000 games yearly. I also must keep track of last-minute assignment changes, make out pay vouchers (which include 1099 tax forms) and supply assignment sheets monthly for 120 officials. I must send contracts out to each league that uses our services and handle all the billing for those leagues. I also inherited the responsibility of keeping an updated mailing label list for our board membership.

All this and more is easily handled on my 1040ST.

I use a variety of software to accomplish my tasks. I use *A-Calc* for game assignment sheets and pay vouchers. For contracts and billing, I use *Word Writer ST* with a calculator desk accessory. When a special type of form is needed, I find myself using *Publishing Partner*. I have made recruitment posters, banquet fliers, etc., with no problems.

I use *Labelmaster* to crank out all my labeling needs. I have also started a small mailing label business using *Labelmaster*.

By profession I am a computer specialist for a large company in Wilmington. As you know, this means I work with IBMs or IBM compatibles. I personally prefer the ST.

—John Gretchen
New Castle, DE

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

## *LARGEST SELECTION IN THE U.S.*

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

## ★ ST SOFTWARE ★

| | | | |
|---|---|---|---|
| Ninja | 14 | ST Gem Programmers Ref Man | 15 |
| Obliterator | 27 | ST Internals Book | 15 |
| Ogre | 27 | ST Intro to Midi Book | 15 |
| Oids | 24 | ST Machine Language Book | 15 |
| Omnires | 23 | ST Peeks & Pokes Book | 14 |
| Orbiter | 26 | ST Pool | 21 |
| Paint Pro | 33 | ST Talk | 5 |
| Paintworks | 14 | Star Fleet 1 | 37 |
| Paperboy | 26 | Star Raiders | 19 |
| Partner Fonts | 21 | Starglider 2 | 26 |
| Partner ST | 46 | Stellar Crusade | 36 |
| Pawn, The | 29 | Stock Market - The Game | 18 |
| PC Ditto | 65 | Strip Poker 2 | 27 |
| Perfect Match | 27 | Sub Battle Simulator | 26 |
| Personal Pascal | 66 | Sundog | 27 |
| Phantassie 1, 2 or 3 | ea 26 | Super Base Professional | 199 |
| Phasar | 59 | Super Cycle | 14 |
| Pinball Wizard | 24 | Super Star Ice Hockey | 33 |
| Pirates of the Barbary Coast | 17 | Swift Calc St | 49 |
| Planetarium | 26 | Tanglewood | 27 |
| Plutos | 21 | Tau Ceti: Lost Star Colony | 11 |
| Police Quest | 33 | Temple of Apshai Trilogy | 13 |
| Power Plan | 52 | Terror Pods | 27 |
| Prime Time | 27 | Test Drive | 27 |
| Print Master Plus | 26 | Three Stooges | 34 |
| Pro Copy | 28 | Thunder | 19 |
| Publisher ST | 79 | Time Bandit | 24 |
| Publishing Partner Pro | CALL | Top Gun | 11 |
| Q Ball | 21 | Trailblazer | 33 |
| Quantum Paint Box | 31 | True Basic | 52 |
| Quink | 11 | Tune Up | 34 |
| Read & Rhyme | 27 | Turbo ST | 36 |
| Renegade | 14 | Typhoon Thompson | 23 |
| Road Runner | 26 | Ultima 2, 3 or 4 | ea 39 |
| Roadwars | 22 | Uninvited | 34 |
| Rockford | 22 | Universal Item Selector | 14 |
| Santa Paravia | 19 | Universal Military Sim. | 31 |
| Scan Art | 33 | Universe 2 | 46 |
| Scruples | 29 | Vampires Empire | 20 |
| SDI | 34 | Vegas Craps | 24 |
| Shadow | 22 | Vegas Gambler | 23 |
| Shadowgate | 34 | Video Titleing | 22 |
| Shard of Spring | 27 | Vip Professional | 149 |
| Shuffleboard | 19 | War Ship | 39 |
| Silent Service | 27 | Wargame Construction Set | 24 |
| Sinbad | 33 | Winnie The Pooh | 16 |
| Sky Fox | 14 | Winter Challenge | 11 |
| Slagon | 27 | Wiz Ball | 11 |
| Soko Ban | 23 | Wizards Crown | 26 |
| Space Quest 1 or 2 | ea 33 | Word Perfect | 239 |
| Spectrum 512 | 49 | Word Up | 49 |
| Speed Buggy | 29 | Word Writer ST | 49 |
| Speller Bee | 29 | World Games | 26 |
| Spiderman | 7 | World Karate Championship | 19 |
| Sprite Factory | 26 | WWF Microleague Wrestling | 33 |
| Spy vs Spy 3 (Arctic Antics) | 19 | Xevious | 19 |
| ST Disk Drives Inside & Out | 18 | 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 #101 ON READER SERVICE CARD.

# ST NEWS

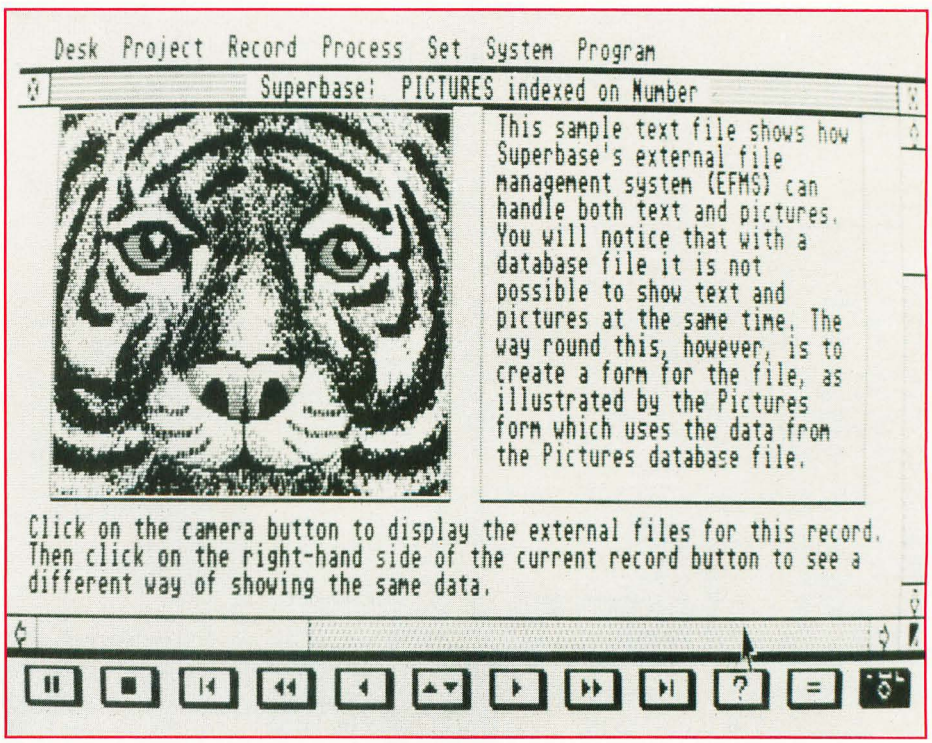
APRIL 1989

## New relational database

The *Superbase Professional* database management system for all models of the ST is now shipping. *Superbase Professional* boasts a forms editor that is used much like a desktop-publishing program to create fully relational, multipage screen forms with cross-file validation, calculation and checking. The forms editor is also compatible with *DEGAS* and *Neochrome* file formats so that pictures generated with those programs may be included in your forms. A resident text editor lets you incorporate ASCII files of any size into your database and provides for key word and pattern searches. *Superbase Professional* is published by Progressive Peripherals & Software, Inc. and is priced at $299.95.

**Progressive Peripherals**
**& Software, Inc.**
**464 Kalamath Street**
**Denver, CO 80204**
**(303) 825-4144**

CIRCLE #130 ON READER SERVICE CARD.




***Superbase Professional*, published by Progressive Peripherals & Software, Inc., is priced at $299.95.**

### Attention MIDI enthusiasts

Casio, Inc. has introduced a new series of MIDI-compatible synthesizers, all reasonably priced so that just about anyone interested in getting on the MIDI bandwagon may do so without putting too great a hurt on their pocketbook. The new Tone Bank keyboards include the MT-240 ($199), the CT-370 ($229), the MT-540 ($269), the DM-100 ($329) and the CT-640 ($399). All the keyboards are 49-key synths, except for the CT-640, which comes with a full-size, 61-key keyboard.

Robert Larson, senior vice president of Casio's Electronic Music Instrument Division, says, "These new Tone Bank keyboards bring to the market some of the most realistic instrument sounds imaginable. The unique layering capability of the Tone Bank gives the user a wider variety of sound choices than we could ever offer in the past."

**Casio, Inc.**
**570 Mt. Pleasant Avenue**
**Dover, NJ 07801**
**(201) 361-5400**

CIRCLE #131 ON READER SERVICE CARD.

**Casio's DM-100 double manual Tone Bank keyboard.**

**Casio's MT-540 Tone Bank keyboard. Features 20 PCM instrument sounds with Tone Bank capability that makes 210 sound combinations possible.**

**Casio's CT-640 Tone Bank keyboard. Features 30 PCM instrument sounds. With Tone Mix capability, 465 sound combinations are possible.**

### An ST compatible?

They say that imitation is the sincerest form of flattery. It's not surprising then that in Germany, where the ST is most popular, one company should be flattering Atari Corp. with its ST-compatible computer. Little information is available on this new machine, but it appears that the Atari-compatible is a suitcase-type machine with two built-in disk drives and a full ST keyboard minus the numerical keypad. The price is unknown.

**IBP Geratebau GmbH**
**Lilienthalstrasse 13**
**3000 Hannover 1**
**West Germany**

CIRCLE #132 ON READER SERVICE CARD.

# NEWS

### HP printer drivers

Neocept Inc., makers of *WordUp*, a GEM-based word processor, and *Fontz!*, a GEM font editor, has released a printer driver for Hewlett-Packard LaserJet and DeskJet printers that will provide quality printouts from most any program that uses GDOS. The *TurboJet* driver comes with instructions for use with *WordUp*, *Timeworks Publisher ST*, *Microsoft Write* and *Easy Draw*, and Neocept claims that the driver cuts print time by as much as 50%. The driver also includes a draft mode that allows rough printouts at five to seven times the normal speed. *TurboJet* requires a one-megabyte ST and a double-sided disk drive, and retails for $39.95.

**Neocept, Inc.**
**547 Constitution, Unit A**
**Camarillo, CA 93010**
**(805) 482-4446**

CIRCLE #133 ON READER SERVICE CARD.

**The suggested retail price of the ICD *FA-ST Tape Backup* is $899.95.**

### Tape backup for your ST

ICD has released its long-awaited *FA-ST Tape Backup*. This cassette tape unit provides for fast, error-free backups of important data, chugging along at a "FA-ST" five to six megabytes per minute and storing 155.7 megabytes on a single data cassette. The software provides individual file recovery and a friendly GEM user interface and can handle *Spectre 128* and *Magic Sac* partitions in image backup mode. To make the system as compact and convenient as possible, the *FA-ST Tape Backup* has room in its case for a built-in hard drive. Drives from 20 to 170 megabytes are available as an option. The suggested retail price of the ICD *FA-ST Tape Backup* is $899.95.

**ICD**
**1220 Rock Street**
**Rockford, IL 61101**
**(815) 968-2228**

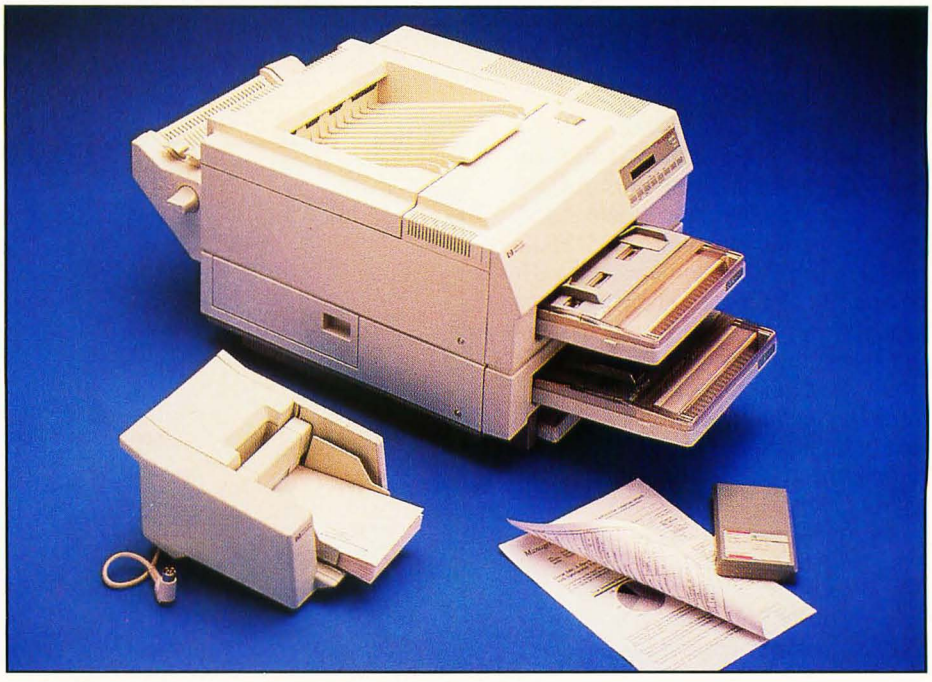
CIRCLE #135 ON READER SERVICE CARD.

### And speaking of HP. . .

Hewlett-Packard has just announced a new printer in its LaserJet family. The HP LaserJet IID is a double-paper capacity, duplex printing printer designed for desktop office use. The IID produces quality images with a resolution of 300x300 dots per inch at eight pages per minute and includes the ability to print on both sides of a sheet. The printer comes with 24 fonts (14 internal and ten on disk), two paper-input trays and can be expanded to higher memory configurations and more fonts. An envelope feeder is also available as an option (for an additional $350). The HP LaserJet IID is available now at a retail price of $4,295.

**Hewlett-Packard**
**3000 Hanover Street**
**Palo Alto, CA 94304**
**(415) 857-1501**

CIRCLE #134 ON READER SERVICE CARD.



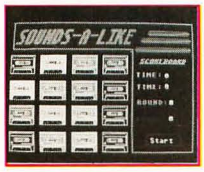
**The HP LaserJet IID is available now at a retail price of $4,295.**

# YOU NEED THE DISK!

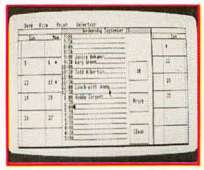
If you want to get the most out of ST-LOG, you're also going to want to get your copy of the disk. Each issue's disk contains all the exciting programs for that issue, including the programs whose listings could not be included due to space considerations. The ST-LOG disk version is truly an excellent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

## ONLY $9.95 EACH!

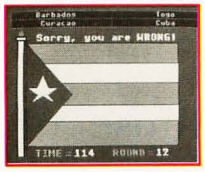
**MARCH 1989** — Picture Puzzle, Sounds-A-Like, ChemCalc And more!



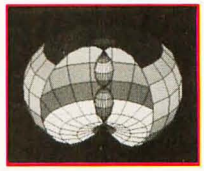

**DECEMBER 1988** — ST Date Planner, Mouse of Fortune, Inside ST Xformer II and more!



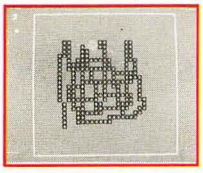
**FEBRUARY 1989** — Flag Trivia, Super Spool, Desk Switch and more!



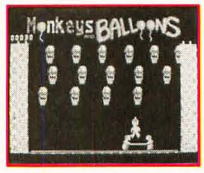

**NOVEMBER 1988** — ThetaTen, Ultra-Graph, Number Maze, And more!



**JANUARY 1989** — ST Date OmniLife, Drama-cide, Transwarp and More!



**OCTOBER 1988** — Monkeys & Balloons, Spectral Sorcery And more!




**DISKS FROM MONTHS NOT LISTED ALSO AVAILABLE!**

## YES! I DO WANT THE DISK! ONLY $9.95 EACH

- ☐ ST-LOG October 1988 Disk
- ☐ ST-LOG November 1988 Disk
- ☐ ST-LOG December 1988 Disk
- ☐ ST-LOG January 1989 Disk
- ☐ ST-LOG February 1989 Disk
- ☐ ST-LOG March 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

# the Animation BRA

BY MAURICE MOLYNEAUX

For those of you who missed last month's installment, this is a new series of articles on using an ST to produce animation. Last month we discussed hardware and the main animation programs. This time, we're on to the act of conceiving a project.

# ation stand:

## N STORMING

### Eureka!

The one question writers/musicians/artists get asked the most seems to be: "Where do you get your ideas?" Some times I'm tempted to reply, "From *The Idea Book* by Cranston Snord," but I usually hold my tongue (no mean trick).

So, before doing anything else on an animation project you need an idea. *Oh sure*, you think, *easy for you to say. Like what?* Well, if I had a foolproof method of coming up with good ideas, you'd all know it because I'd have already written a best-selling book on the subject and retired to Barbados. Sadly, there's no pat and easy answer. I'm not much of a believer in inspiration, especially not the "bolt out of the blue" variety. In my experience, ideas don't just happen; like just about every other thought in our brains, they come forth through the mind's processing of information already received.

Some philosophers say there's "no such thing as a wholly original thought," meaning that all ideas are extensions and modifications of those that preceded it. I can't say that this is entirely accurate, but it does hold a grain of truth. I've found that the more I experience and learn, the better I am at coming up with ideas; concepts pop readily into my brain, and what's more important, they are *diverse* ideas. I do not find myself locked into one genre.

For example, my extensive studies on classic short-subject animation were instrumental in my deciding to make the *Art & Film Director* sales video in the form of a funny cartoon. Had I not had the knowledge I do about these kinds of films, the idea probably would not have occurred to me, and I would have made something completely different. Another project I am working on is a serious animated music video with realistic human characters, one presented in the form of a short film that actually could be understood without the music it is designed to accompany and accent. (But it's better *with* the music, of course.) This is a complete turnaround from the cartooniness of the *Art/Film* project, and one I would not have been able to do it had I been the type to focus entirely on one type of film.

I should point out that focusing on one particular medium or style is not bad. What I am saying is that the broader your background, the more diverse a range of material you will be familiar with, and the better equipped you will be to get the most out of the medium you have chosen for a particular project. Even the serious films usually have some light moments and funny films their serious times. To carry these off effectively you have to know about a lot of subjects or be willing to research them.

So, the best thing to do is to think about the kinds of things you'd like to animate, pick a genre (i.e., funny cartoon, space opera, etc.), and do some in-depth research on it. Study films that use the subject, and, even when you look at films, books, and other mediums not related to the subject (you can learn a lot about storytelling by studying novels and scripts, or about visual styles from paintings, books on anatomy, photographs and advertising art), look at what they do and try to understand why they have done things in certain ways. Look not only for good things, but also pay close attention to what you feel are errors. And, when you find such a mistake, think of a way it could have been fixed. In this way, you build up a library of references in your head, good and bad, and can use these as a wellspring of ideas *and* a checklist.

For example, if you noticed that certain types of shots in films tend to be boring, you'll want to avoid using such a shot—unless you want to use that effect. For example, you might use a slow, stately pan to lull viewers into a false sense of security before springing some surprise on them. What you might perceive as wrong may be wrong only in the context it was used. Likewise if you were very impressed with a particular visual effect, you might find that it works well only in the context you saw it and not in the project you are planning. The point is that good and bad are (generally) relative terms film/animation-wise and not absolutes. You'll have to learn to judge things based upon their application in a given situation, not on a set of absolute standards.

In case you haven't caught on yet, what I'm trying to impress on you here is one of the ground rules of critical thinking—and one of the most difficult things to apply. You have to be willing to reject ideas that you like, but which are not appropriate to your subject. And at the same time be open to using things you don't particularly fancy, but which would work to your advantage in the context of what you are trying to do.

At this point, you might wonder why I haven't discussed software. Well, I tend not to consider that in my initial concept of a project. I prefer to come up with the

ideas unhindered by the limitations of the software and/or hardware at my disposal. After I determine what I would ideally want to do in the animation, I get down to the nuts and bolts of how to do it and which program, or combination thereof, would be best for the task. As I have just about any paint/animation tool there is for the ST, this is not much of a problem.

If you are using only one type of program, clearly that will limit your choices a bit. You may then have to take in account the kinds of things you can do with the software at hand. However, I advise you not to think on this too much, for being aware of the limits of your tools during the planning stage may deter you from a good idea, because you are afraid your computer and software are incapable of the task. That cliche "Where there's a will there's a way" perfectly sums up my feelings on this.

## Topicality

Okay, so you should start thinking about a topic. Just think of something that you'd like to animate. Don't try to work out all the details or even consider if it's possible. Just try to form a general idea of the type of thing you'd like to do. I'll think of one too. It doesn't matter what the subject is or how long you plan the animation to be. It could be a one-shot ten-second gag or a ten-minute movie. I do suggest, however, a short animation. No more than a few minutes long, so you can actually finish it in some reasonable time. Got your concept? If not, think a bit more and come up with the idea before going on to the next paragraph.

Got it now? Good. The type of project I'll pick for myself will be. . . oh. . .a music video, one that is presented in highly stylized cartoon form. Is my idea a bit more complicated than yours? Probably; but because it's a complex project, it will be a good one to use to illustrate the creation of a project from the ground up. Why? Because not only is there the problem of linking appropriate images to pre-existing music, but there are timing problems, questions of graphic style, pacing, et al. Now, don't shift gears on your own idea and try to do what I am going to be doing, because if you're going to be creative you have to start doing so right now and not emulate what I'm doing. Just apply the examples I give to what you want to work on.

## Breakdown and brainstorming

The first thing to do is look at the idea and start considering how to do it. In my case, the first thing I'm going to need is a song to animate the video to. I decided to animate a video to the song "I Know You" by the band Catzeye. What? You never heard of them? Well, not surprising, really, as the song in question is from an (as of this writing) unreleased album. Catzeye is a band whose members include ST-LOG's West Coast editor Charles F. Johnson, as well as Jim Studer, George Hawkins and Tris Imboden (all four are professional musicians who have appeared on the albums of and toured with many well-known artists—such as Al Jarreau, Kenny Loggins, Stanley Clarke, etc.). One of the songs on the album seemed a suitable subject for the type of animation I want to try. Not only is it a good song, but as I've known Charles for a while, getting the rights to use it was easy.

(If you chose something musically oriented, you could use a song you think appropriate. However, if you *do* animate to it, be forewarned that you cannot really do anything with your animation other than show it to people (non-commercially) because you probably don't have any legal rights to use the music, unless it's your own music or that of a friend who gave you written permission. If you animated to a Todd Rundgren tune and tried to sell your work without first obtaining rights to the song. . .well, does the word "lawsuit" mean anything to you?

Now is about the time to whip out some pencils and blank paper and get ready to sketch. Most good animators "script" with sketches. Sketching is a useful tool because it not only helps you record your thoughts visually, but it also allows you to see on paper what you are picturing and explore it. An image that seems crystal clear in your head often seems impossible to get down on paper. Usually that's because it's not as clear in your head as you thought it was.

For example, I tend to mentally act out the action I'm planning, but it's kind of like seeing the action from the point of view (POV) of the character or object that is going through the motions, and not from the third-person perspective of the "camera." The action is clear in my mind, but the actual appearance of it from a specific angle is not. In a like vein, I may have an image of the "look" I want for the characters and settings, but getting that on paper is also tough.

As you think, sketch. Don't worry if your drawings aren't detailed or even in any kind of order. Just scribble down a thumbnail sketch of any ideas you have. Don't wait until you have a clearer idea of what you're doing. You're not wasting time by drawing what you're thinking. These sketches can help you build a clearer idea because you can *see* what you're thinking. Also, by drawing these, you don't forget them as easily (easy to do on any decent-size animation project). If the ideas are coming so fast and furious that you can't sketch quickly enough to keep up with your concepts, and you are afraid of losing some of the ideas while you are busy drawing, quickly jot short notes along the edge of your sheet. You can then go back to sketching without fear of totally forgetting an idea. Sometimes a single word is enough to jog your memory back to what you were thinking.

Probably the best technique for coming up with ideas is a brainstorming session. Get together with your partner(s), or someone who is interested in what you are doing, and just kick ideas around. One of the most important rules in these types of sessions is to not shoot down ideas. Not only might you be completely closing off one approach you might find useful, but you tend to wreck the creative atmosphere by knocking down another person's concepts. Not much more to say on that, just keep notes on the ideas you come up with (or tape the conversation), so you can peruse them later.

One of the best places for brainstorming I have found is in a conference room on a telecommunications network such as DELPHI. Not only can you brainstorm with several people from distant locations at once, but if you are using good telecommunications software, you can automatically keep a transcript of everything said by using a capture buffer. The problem with this method is that it requires each person to have a computer, a modem and be a member of the service in question, not to mention the charges for using the system. Still, they're good places to work, if you can use them.

Another good method of idea-generating is to lay out what you want to do in your animation and determine how it starts and where it ends. An example: In the *Art & Film Director* video I created for Epyx, there was to be a scene that was a parody of a Japanese cartoon—the kind with giant robots and such. The animation was to make *fun* of these cartoons, so I studied some of them. I made notes of some typical cliches of the medium:

—Characters have almond-shaped eyes three times the normal size, and mouths one third the normal. They have very simple features, pointed chins and unruly mops of hair.

—Backgrounds are often lavishly detailed, often overloaded with small details. Robots likewise.

—Use of "modeling" (shadows, particularly on faces) on characters.

—Lots of camera work. Viewpoints change a lot. Also lots of pans (horizontal camera movement) and zooms. Lots of close-ups on characters.

—Often there is a child character, usually a boy, who thinks he's tough. The kid is usually short, kind of round, has rounder eyes than the "straight" characters, and usually possesses a *big* mouth and a highly obnoxious voice that grates.

—Animation is choppy, probably running only a few frames per second, as opposed to the 12 to 24 frames per second of "full" animation.

—More care lavished on flashy effects sequences than movement of people. Character animation (giving characters distinctive types of movements that help convey their attitudes and personalities) is nonexistent.

—As dialogue is looped (dubbed/rerecorded) with English-speaking actors replacing the Japanese voices, the dialogue tends to be out of synchronization with the mouth movements (which tend to be mere open and closed positions only).

Now, on this foundation I had to build my sequence. My initial picture was of two "straight" (a film term meaning "normal" non-comedic) characters aboard a giant robot spacecraft, conversing about some problem. Momentarily, a kid character in the background starts talking tough, rattling on and on until we can stand it any more. At this point, Megabit Mouse (the main character in the *Art & Film* video) enters the scene and shoots the kid to shut him up.

Why did I choose this action? I wanted to parody this medium, so I planned to use and exaggerate the cliches I noted. Showing a detailed giant robot that moved smoothly would contrast with the choppy movement of the characters. The straight characters would be typical Japanimation types (a popular term for this type of cartoon) slender of build, with giant eyes, tiny mouths and messy hair. They would contrast with the kid, who, as befits his stereotype, would have a truly obnoxious voice and a *big* mouth. Where a Japanese cartoon has choppy movement, I would make it choppier. Where voices were out of synch and annoying, I would exaggerate those. You have to be true to the medium, even when you're making fun of it, otherwise your audience may not recognize what you're doing.

At this point I scribbled out a few sketches to establish something of the look I wanted. That didn't take much, so then it was time to go to the storyboard.

### Oops!

We've run out of space! We'll continue our discussion next month, where I'll be writing about storyboarding and specifically about camera angles, breaking down action into scenes and viewpoints, visual linking devices, ad infinitum. I'll also be showing you the actual storyboard for the Japanimation sequence described above. See you then. ■

---

*Maurice Molyneaux began playing with stop-motion model animation in high school and started working with computer animation on his 800XL using* MovieMaker. *Using his STs, he has created animated presentation and sales videos for software publishers, and co- authored two Cyber Studio design disks with* ST-LOG *associate editor Andy Eddy. He continues to produce aftermarket products for ST animation packages and consults on animation systems, as well as continues work on various video-related projects.*

# C-manship

**Performing animation on the ST is a little bit tougher than it is on the 8-bit Atari computers, if for no other reason than the ST does not have player/missile graphics. The programmer is responsible for every step of the animation, getting very little help from the hardware itself. Even so, coming up with a simple animation sequence is not particularly difficult—as long as you are willing to do some preliminary work and are competent in handling MFDBs and raster operations.**

**In this installment of *C-manship*, we will study the creation of an animation sequence through each step of the process. But we will not review previously covered material; therefore, if you are not comfortable with MFDBs and the *vro_cpyfm()* function, I strongly advise that you review the *C-manship* on raster operations (Issue 12) or find material on that subject in your other reference materials.**

## The program

Listing 1 is this month's sample program. It was created using *Laser C*, the new version of *Megamax C*. But don't worry; with one minor change it will compile and link fine with the older version of *Megamax C*. Simply change the *FDB* in the function *draw_icon()* to *MFDB*.

For those of you who are planning to switch to *Laser C*, a word of caution. The header files that come with the package have been modified to be more compatible with the "standards" that have been set up for GEM and the ST. The FDB-to-MFDB change is just one of many changes, and when you first try to compile a program created with the older *Megamax C*, you may find that you are going to have to juggle your *#includes* a little to get the code to compile properly.

Now back to our sample program. Once you have the program compiled and linked, go ahead and run it. The screen will go blank, after which you should press the left mouse button. A spaceship will appear on the right-hand side of the screen, and a photon missile will start moving toward it from the left. When the missile collides with the ship, the ship will explode, and you'll once again be faced with a blank screen. Either press the left button to see the animation again, or the right button to exit back to the desktop.

## The first step

Before we program an animation, we must, of course, have something to animate. In other words, the first thing we must do is take out a program like *DEGAS* and draw the various figures that will make up the animation sequence. Each figure in the sequence will be slightly different than the one that came before, moving us closer and closer to the final result.

For example, if we wanted to have an exploding ship, we would start with the ship itself as Frame 1. Then we would take that ship and add a yellow glow to its center; this would be Frame 2. For Frame 3, we would expand the yellow glow. Frame 4 would show the ship completely engulfed by the glow, and in the last frame we would have the ship disintegrating into pieces. If you take a look at Figure 1, you can see this process. (The sprites shown were created by Maurice Molyneaux for "Moonlord ST" from the

# ANSHIP

## SIMPLE ANIMATION

BY CLAYTON WALNUM

July '88 issue of ST-LOG. The ship drawings are slightly different from the ones used in this month's sample program; the original drawings fell victim to a hardware failure long ago.)

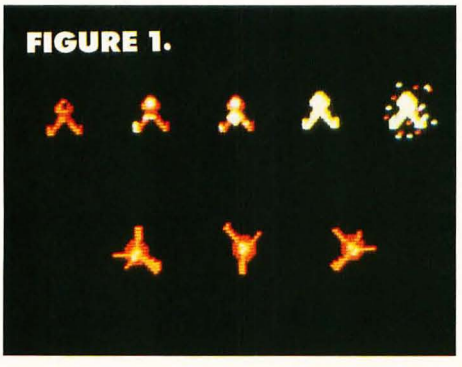
In our sample animation, we have a photon missile moving across the screen and hitting the alien ship, causing it to explode. The three frames of animation for the photons are also shown in Figure 1.

Once we have created all the figures we need for the animation, we must convert the graphics into numerical data. We do this using a sprite editor, such as *Raster Sprite Editor* from Issue 16 of ST-LOG. These editors allow the user to "cut out" a section of a picture and will automatically convert the cut-out section into the proper form of data. All we have to do then is place the data into our program.

If you look at the top of Listing 1, you will see the graphics data for our sample animation. The data labeled *alien[ ]* is the alien ship. The data blocks labeled *expl1[ ], expl2[ ], expl3[ ]* and *expl4[ ]* are the alien ship in its various stages of disintegration. Finally, the blocks of data labeled *photon1 [ ], photon2[ ]* and *photon3[ ]* are the figures for our photon animation sequence.

### Programming the animation

Now that we have all our figures converted to data, we must come up with a program that'll move that data on and off the screen in such a way as to create the actual animation. In the case of the photon, we must copy each figure to the screen in sequence, while at the same time moving the photon toward the alien ship. The exploding alien ship will be a little easier to do, since we don't have to move the ship itself.




Let's take a look at the program listing. First turn your attention to the function *main( )*. Here we initialize the application and open a virtual workstation. Then we check the resolution to make sure that the user is not trying to run the program in medium or high resolution. If we're in the wrong resolution, we bring up an alert box that informs the user of his error, after which we close the workstation and exit back to the desktop.

If the screen resolution is okay, program execution goes to the function *do__animate( )*. In this function, all we really do is monitor the mouse buttons. If the left button is pressed, we perform the animation, then come back to wait for another button press. If the right button is pressed, we exit the program.

### The photon

When the left button is pressed, program execution jumps to the function *photon( )*, where the actual animation begins. In *photon( )* we first store the address of each image of our photon animation into the array of pointers, *ph[ ]*. We then turn off the mouse and draw the image of the alien ship on the right-hand side of the screen using our own function, *draw__icon( )*. (This function is a slightly improved version of the *draw__icon( )* that appeared in Issue 12's *C-manship*.)

Next we initialize the X and Y coordinates of the photons, as well as the color stored in *pen*. The *while* loop that follows this initialization will repeat until *pen* is changed to one of the colors that make up the alien ship. Within the loop, we draw each stage of the photon animation, moving it slightly to the right each time.

To animate the photon, we first draw the initial figure in the sequence, using *draw_icon( )*. Then, in order to control the speed of the animation, we call the AES function, *evnt_timer( )* like this:

```
evnt_timer ( low, high );
```

Here, *low* is the low word of the number of milliseconds to pause and *high* is the high word of the number of milliseconds to pause. This function always returns a 1.

After the pause, we redraw the sprite in the same position. Because we are using a writing mode of 6, which Exclusive ORs the source and destination values, this second drawing of the sprite erases the first from the screen, leaving the screen exactly as it was before we drew the sprite. The only problem with this method of drawing a sprite over a background is that when the sprite is first drawn, it will appear transparent; that is, any graphics behind it will show through. There are ways to overcome this problem, but we'll save that for a future installment of *C-manship*. In this case, because our background consists of nothing more than a black screen, we don't have any of the problems associated with Exclusive ORing a sprite with the background.

After erasing the first sprite, we add 4 to *x*, the sprite's X coordinate, so that the next photon sprite will be drawn farther to the right, closer to the alien ship. After that, we increment *p* so that the next time we use *p* to index our array of pointers, *ph[ ]*, we'll be pointing to the next image in the photon sequence. Finally, we get the color of the screen at our current location and call *evnt_timer( )* to pause 50 milliseconds before going on to the next frame of the animation.

To retrieve the color of the pixel at our current screen location, we use a call to a VDI function, *v_get_pixel( )*, like this:

```
v_get_pixel ( handle, x, y, &pixel, &pen );
```

Here, *handle* is the virtual workstation handle, *x* and *y* are the X and Y coordinates of the pixel we wish to examine, *&pixel* is the address of an integer that will be set to 1 if the pixel being examined is set and set (unset?) to 0 if it's not, and *&pen* is the address of an integer that will contain the color value of the pixel being examined. If the color detected is one of the colors that make up our ship (in this case, we're looking for RED or DRK_RED), we know that the missile has collided with the ship. We drop out of the loop, after which program execution jumps to the function *kill_alien( )*, where we will perform the exploding ship animation.

**Simple animation on the ST is not particularly hard. However, you should be aware that, unlike its 8-bit little brother, the ST is not really well designed for animations involving many objects at once.**

## Kaboom!

Because we don't need to actually move the ship, the explosion sequence is much easier. All we have to do is draw the four frames of the animation one after the other, providing a short delay between them and erasing each image before drawing the next. This time around, we're erasing the images by drawing a black circle on top of them, rather than using the Exclusive OR method described above.

Once the main sequence is completed, we drop into the code that draws the "sparkles" on the screen—just to add a little pizazz. Here we're doing nothing more than using *v_pmarker( )* to draw 40 crosses in random locations within the area that the ship occupied. Because we're drawing them so fast, you would swear that there were many of them on the screen at the same time!

When we're through in *kill_alien( )*, it's back to *photon( )* to turn the mouse back on, and then back one more step to *do_animate( )* to wait for the next mouse button click. A left click will repeat the animation, while a right click will cause us to exit the program.

## The end again

As you can see, simple animation on the ST is not particularly hard. However, you should be aware that, unlike its 8-bit little brother, the ST is not really well designed for animations involving many objects at once. The lack of player/missile graphics makes programming arcade-type action games a real chore. And there's really no way to do it in C. You have to have the speed of assembly language.

Still, the techniques presented here can be effective and fun in simple applications. Experiment a bit, and see what you come up with. You might surprise yourself.

# Attention Programmers!

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

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

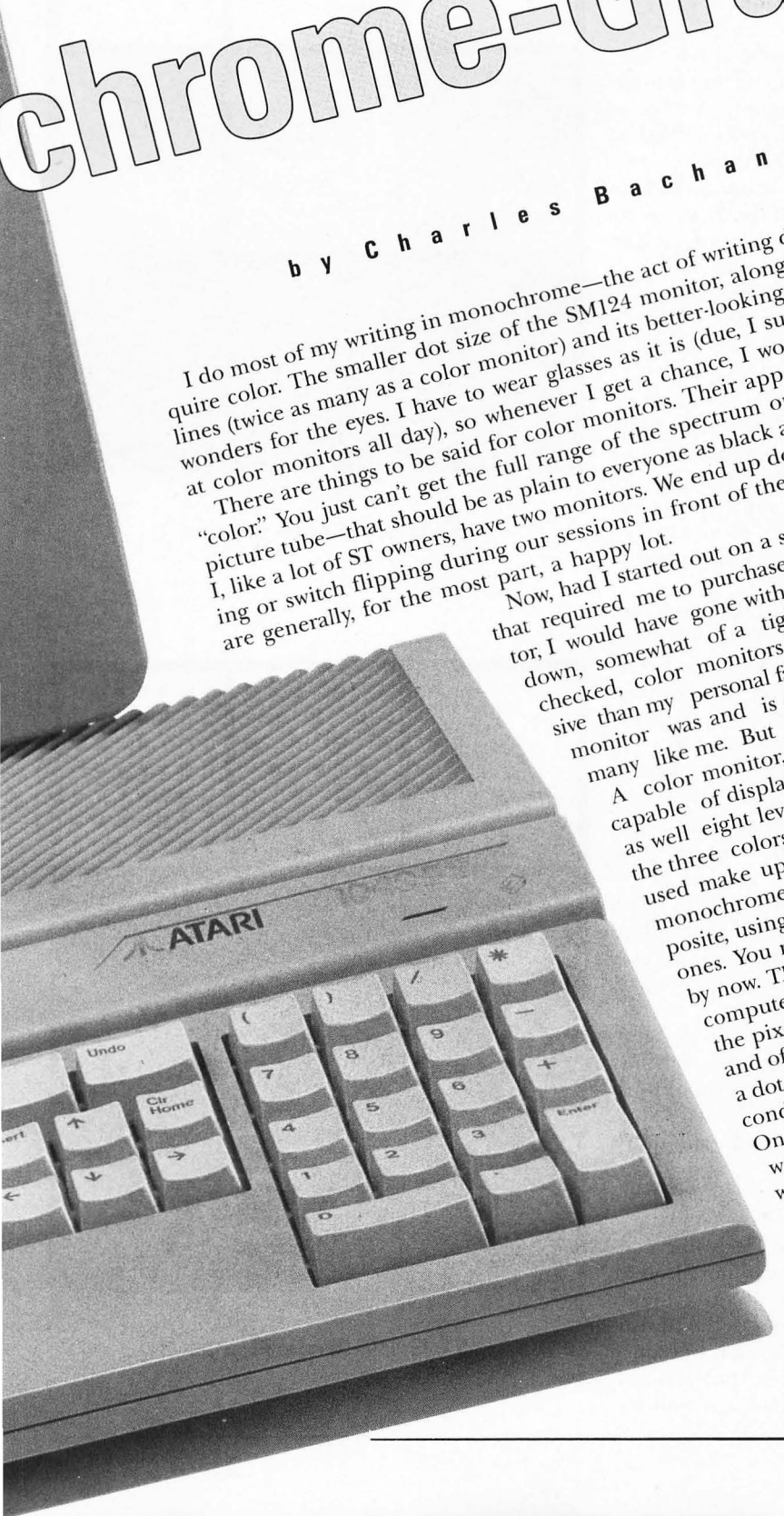
Send your programs and articles to:
ST-LOG Magazine
P.O. Box 1413-M.O.
Manchester, CT 06040-1413


Mono
ATARI
SC1224
Esc
Tab
Control
Backspace
Delete
Return
Shift
CapsLock

# chrome-Gray

**by Charles Bachand**

HIGH RESOLUTION ONLY




I do most of my writing in monochrome—the act of writing doesn't usually require color. The smaller dot size of the SM124 monitor, along with its 400 scan lines (twice as many as a color monitor) and its better-looking character set does wonders for the eyes. I have to wear glasses as it is (due, I suspect, from staring at color monitors all day), so whenever I get a chance, I work in monochrome.

There are things to be said for color monitors. Their appeal lies, of course, in "color." You just can't get the full range of the spectrum out of a monochrome picture tube—that should be as plain to everyone as black and white. That is why I, like a lot of ST owners, have two monitors. We end up doing a lot of plug pulling or switch flipping during our sessions in front of the ST's keyboard, but we are generally, for the most part, a happy lot.

Now, had I started out on a somewhat limited budget that required me to purchase one and only one monitor, I would have gone with monochrome. I am, deep down, somewhat of a tightwad, and the last time I checked, color monitors were much more expensive than my personal favorite. Yes, a monochrome monitor was and is the way to go for me and many like me. But there is something missing.

A color monitor, being an analog device, is capable of displaying a wide range of colors, as well eight levels of brightness for each of the three colors (red, green and blue) that are used make up what we see on the screen. A monochrome monitor is pretty much the opposite, using digital signals instead of analog ones. You must all be familiar with "digital" by now. Those ones and zeros that run your computer are also responsible for turning the pixels on your monochrome screen on and off. Force a bit on (a "1" condition) and a dot lights on the screen. Shut off a bit ("0" condition) and a pixel is extinguished. Ones and zeros and that's basically it. So, with only two possible states, there is no way to produce any shades of gray. Either a pixel is white, or it's black. There is no middle ground to tread. Or is there?

## The Pixel Gazette

The Sunday newspaper—how often have you looked at it? For those who manage to make it past the comic section, there are usually quite a few articles with photos associated with them. Photos of escaped criminals, of people running for office, of children with dripping ice cream cones, of cute and cuddly puppies. If you take a good look at these black-and-white photos with the help of a magnifying glass, you'll see that they're made up of tiny dots. They almost remind us of pixels, don't they? In fact, they are pixels not dissimilar to those on the ST. Our computer's pixels are arranged in neat horizontal and vertical rows and columns, while the news photos use a crosshatch pattern.

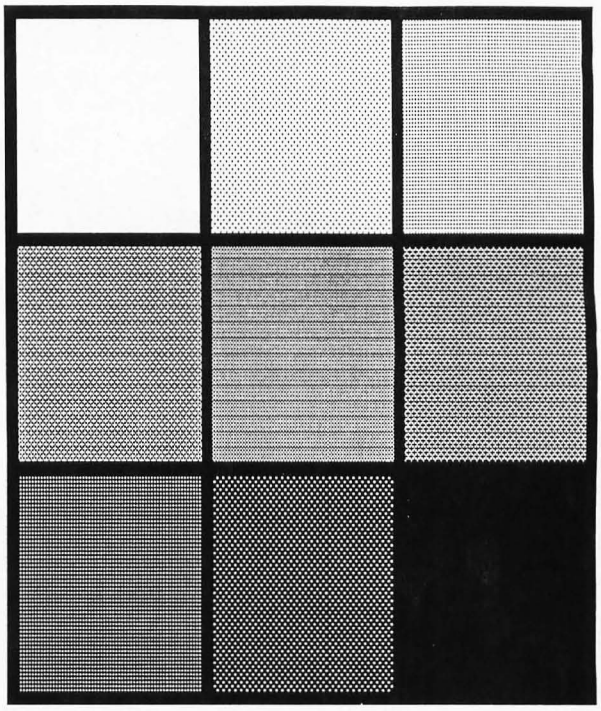
If held at a normal distance, that photo accompanying the article about the increasing price of liverwurst in the home and garden section exhibits a characteristic that is lacking in the black-and-white world of monochrome—shades of gray. The clouds are white, and the asphalt is black (this is a photo of liverwurst?) with just about everything else falling somewhere in the middle. These shades run the full gamut from light gray to near black to just about every gray imaginable. If the *Times* or the *Wall Street Journal* can do that with pixels, why can't we? Well, luckily we can.

We can accomplish this little feat of magic by not using all the pixels in a given area at once. To achieve a 50% brightness level, we merely turn on half the pixels on the screen and turn off the other half. This can be considered analogous to only turning on half the lights in your house at night. Naturally, it depends which lights are lit as to what visual effect is achieved. If you were to turn on all the lights in the top floor and extinguish all those on the bottom, we would certainly fulfill the above stated rule—the only problem being is that the effect is all wrong. Pulling this little trick with the house on your ST's monochrome monitor will produce a 50% gray effect that will look good to someone standing at a distance of 200 feet perhaps, but at the normal range of one or two feet, it will look like a white rectangle on top of a black one. What we need is a gray-scale effect that will look good close-up.

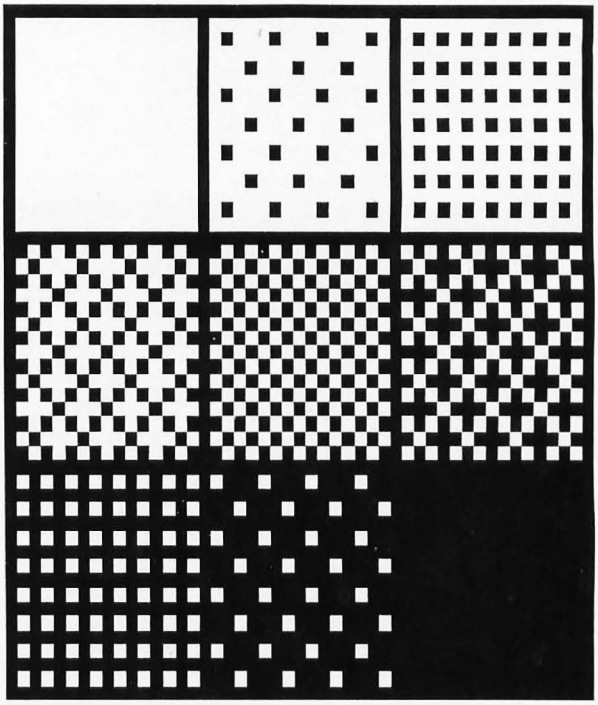
And what do we find buried deep in the ST's operating system, but a gray-scale that looks good close-up. Well, relatively good anyway. When 640 pixels are mapped onto an eight-inch-wide screen, this gives us a resolution of only 80 DPI (dots per inch). This is far less than what appears in newsprint and what is produced by most laser printers that generally draw text and graphics at 300 DPI. But even laser printers tend to seem coarse when compared to expensive Linotype machines, that can handle 1,200 and some even 2,400 DPI. Impressive. But, until we hit it big with the lottery or inherit an oil well or something, we'd better pull our heads out of the clouds and get back to the affordable world we live in—the world of the ST.

Figure 1 shows an example of the various levels of shading used by GEM, while Figure 2 presents a magnified view of the pixel patterns used. This is one way (actually, the most common way) of simulating shades of gray in monochrome. But it is not the only way, and it is certainly not the way we will be doing it.

We shouldn't need to examine an area of the screen to discern gray, but instead should be able to get that information from one and only one pixel. We should not see a pattern that looks gray; we should just see the gray. That's what we get with color monitors, and that's what I hope to accomplish here through a quality of the phosphors in the monitor's picture known as persistence.



**ST Gray-Scale Figure 1**



**ST Grey-Scale Figure 2**

## A bright idea

Ever wonder how a light dimmer works? Those little electronic marvels that replace wall switches are showing up in more and more homes. They are small, cool running (the dimmer, not the bulb) and highly efficient—but it was not always this way. Before the advent of the transistor, one needed a very large, mechanically controlled rheostat to reduce the amount of current going through a light bulb. These rheostats also tended to generate quite a bit of heat; so mounting them in a wall was

asking for trouble.

Nowadays, things are much simpler, and it's all because of an small electronic component known as a Triac. A Triac (the General Electric trade name for the device, by the way) is nothing more than a very fast, bidirectional, electronic switch. When wired in series with your common, garden-variety 100-watt light bulb it does an amazing thing—or rather, it is simply amazing what it doesn't do! It does not reduce the voltage or the current going into the bulb.

Instead, the Triac turns the bulb on and off very quickly, at a rate of 120 times a second. This switching effect is timed to synchronize with the up and down voltage swings that appear in your everyday 60-cycle AC house current, as illustrated in Figure 3A (the lamp operating at full brightness). Figure 3B shows the voltage present at the lamp with the dimmer circuit set to half brightness. The bulb has only enough time to reach the brilliance set by the dimmer when the power is abruptly turned off. It does this again and again 120 times each and every second. The bulb itself takes far longer than the $^1/_{120}$ of a second to visibly go from off to on and then off again. The thermal "persistence" of the filament provides the stability needed so that we don't all go bonkers from the flickering.

**Figure 3A—Full Brightness**

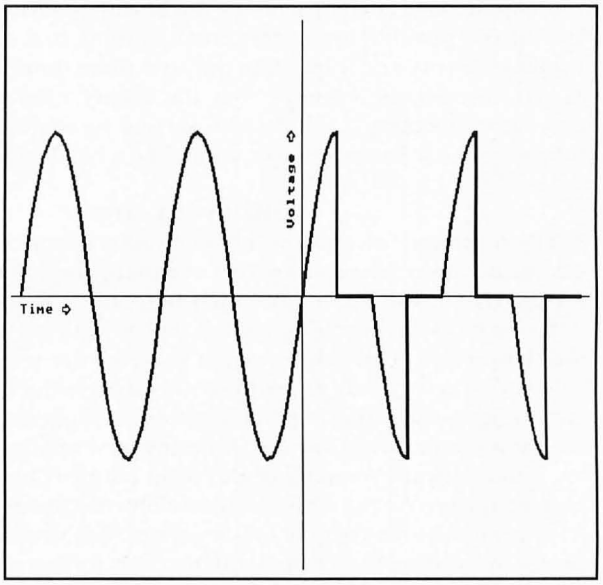


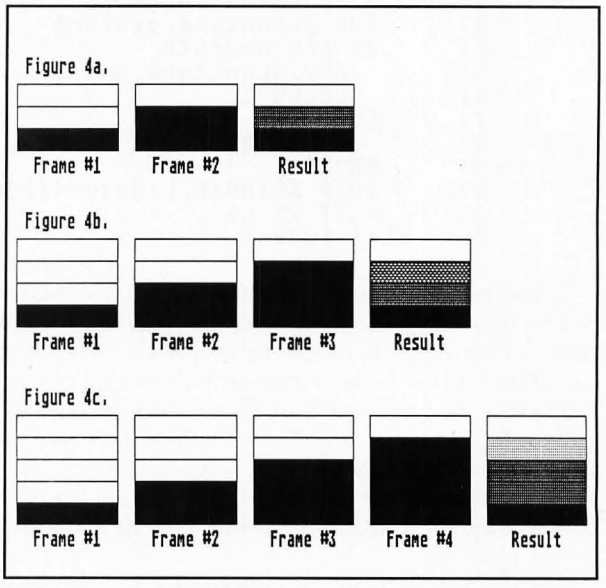

**Figure 3B—Half Brightness**

The phosphor material found inside a fluorescent lamp also exhibits persistence, though not as dramatically or for such a length of time as that of an incandescent light. Its operating parameters are not related to temperature. Rather, a fluorescent lamp works by passing a high voltage through a gas-filled tube that generates long-wave ultraviolet light. These ultraviolet rays strike the phosphor material that coats the inside of the fluorescent tube. The phosphor absorbs the UV light, thus causing it to glow with visible light. Oh, in case you're wondering, those "black light" bulbs that you see used at Halloween parties or in window displays are merely fluorescent lamps that someone forgot to coat with phosphor.

Now what does all this stuff about light bulbs have to do with monochrome monitors? Well, the interior surface of the picture tube found inside your monitor is similar to the internal makeup of a fluorescent light bulb; the difference being that the phosphor on the inside of a picture tube glows not because it is being hit with ultraviolet light, but rather from a head-on collision with a beam of electrons. The effect is the same in both cases: The phosphor gives off visible light and takes time to stop glowing once the source of excitation (either the ultraviolet light or the stream of electrons) is extinguished.

## Just like in the movies

When we look at a monochrome screen, we are presented with a still image that is being flashed at us 70 times each second. If we were to take out every other frame so that we were presented with the sequence PICTURE, blank screen, PICTURE, blank screen, PICTURE and so forth, we would still see the original image but something strange will have happened to it: It seems dimmer. The phosphors are now given a chance to darken between frames of information. You might even notice a slight flickering effect. If we inserted two blank screen frames between our actual images, the sequence would seem darker still and the flickering effect more pronounced. The more blank screens, the darker the apparent image and the more pronounced the flickering.




We can change the mixture of blank screens to data images to favor the data. PICTURE, PICTURE, blank, PICTURE, PICTURE, blank, etc., will seem brighter than the 50/50 mix originally tried, though it's not as bright as when we had no blank screens at all. Have you noticed that we can increase the number of different apparent brightness levels by increasing the number of frame sequences shown before repeating?

Since we want to have more than one level of gray on the screen at once, it would be silly to just insert "blank" frames. If we want graphics near the top third of the screen to be white, the middle third gray and the bottom black, all we really need are two alternating screens as shown in Figure 4A. Both frames show white at the top and black at the bottom

with the middle third alternating. By merely adding extra frames to the sequence as shown in Figures 4B and 4C, we can generate three and four visible brightness levels.

Each frame will be taking up 32,000 bytes of memory in your computer. A seven-frame sequence that will produce eight noticeable brightness levels (numbered 0..7) will require 224,000 bytes of free RAM to store. It will take $^1/_{10}$ of a second to show these seven-frames before repeating, thus producing a now very noticeable flickering effect. A whopping 672,000 bytes would be needed to generate the 21 different frames needed to simulate all the possible brightness combinations that a color monitor can produce. Twenty-one frames not only use three times as much RAM space as our seven-frame example, but the flicker rate is three times more pronounced, taking a full $^3/_{10}$ of a second to come full circle. It can no longer be classified as a flicker; more like a headache waiting to happen!

## Dithering time

The technique of drawing noncontiguous frames until the specified gray value was reached was what I originally used and planned on using for the GFA BASIC code presented here; but I found a way to improve it that reduces the perceived level of flicker involved without really changing the picture. This improvement involves the use of dithering.

Now don't everybody jump down my throat with "But you said you weren't going to use dithering or fill patterns!" That's true, I'm not—not in the conventional sense anyway. Dithering, as it applies to computer graphics, usually means scrambling the pixel patterns in order to generate a blurred image. We normally use these blurred images (fill patterns, really) to generate tones of gray on the screen. But what if we don't scramble pixels? What if we dither the sequence that frames appear on the screen instead? Remember, I'm trying to blur the flickering effect and Figures 5A and 5B show what happens when we do that. Note that "time dithering" greatly reduces that flicker rate for grays near the middle of the scale.

The algorithm used is quite simple, as the following block of GFA BASIC will show:

```
1:   gcounter&=0
2:   FOR index&=1 TO gmax&
3:     SUB gcounter&,gcolor&
4:     IF gcounter&<0
5:       ADD gcounter&,gmax&
6:       COLOR 1
7:     ELSE
8:       COLOR 0
9:     ENDIF
10:    VOID XBIOS(5,L:gptr%(index&),L:-1,-1)
11:    PLOT x&,y&
12:  NEXT index&
```

Line 1 initializes our accumulator variable *counter* to 0. Lines 2 and 12 define the limits of our FOR/NEXT loop that will take us through the different drawing frames starting with No. 1 and ending with the last frame as defined by the variable *gmax&*. We subtract our desired gray value as defined by the variable *gcolor&* from our working accumulator *gcounter&* in Line 3. In Line 4, if *gcounter&* is less than zero, we add the value of *gmax&* to it and set the drawing color to 1 as in Lines 5 and 6; otherwise, set the drawing color to 0 as seen in Line 8. Line 10 uses an XBIOS call to select the logical drawing frame area in RAM, and Line 11 does the actual plotting.

All quite simple, really. It is interesting to note that when this algorithm is modified slightly, it is the one used by the operating system to draw diagonal lines on the screen. We'll delve deeper into the working of these routines a little later on.

## Lights, "camera," action!

So while the flickering has been greatly reduced, there is still some that can be seen, especially for very bright and very dim shades of gray. "So

**Figure 5A**

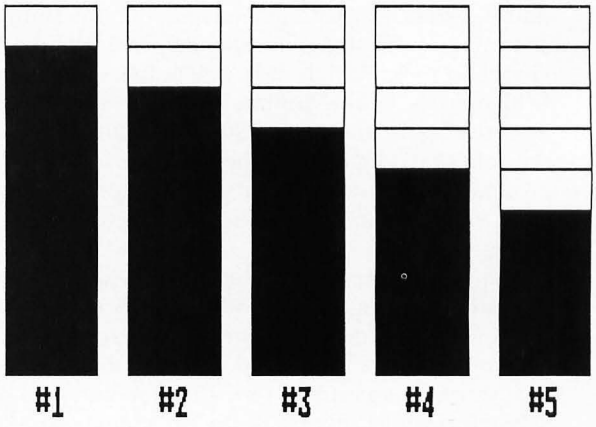
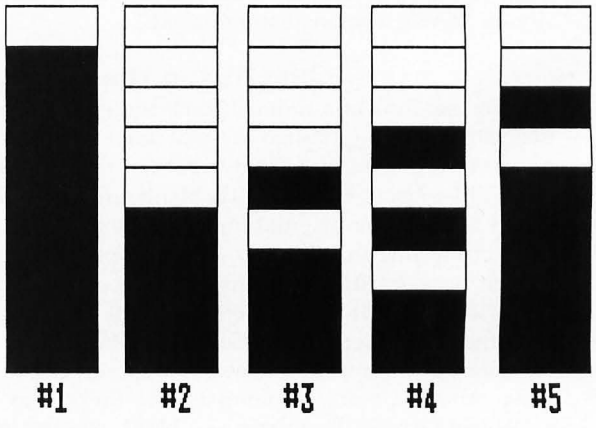
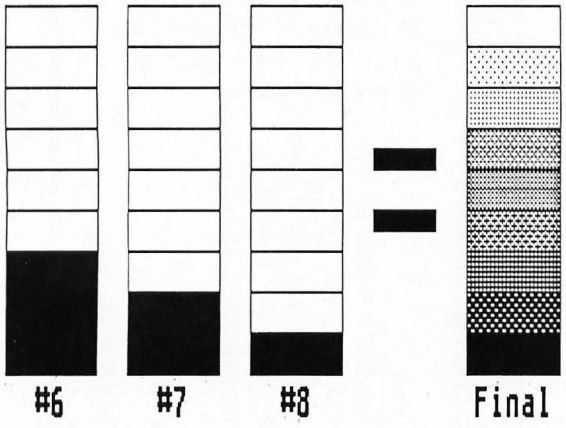
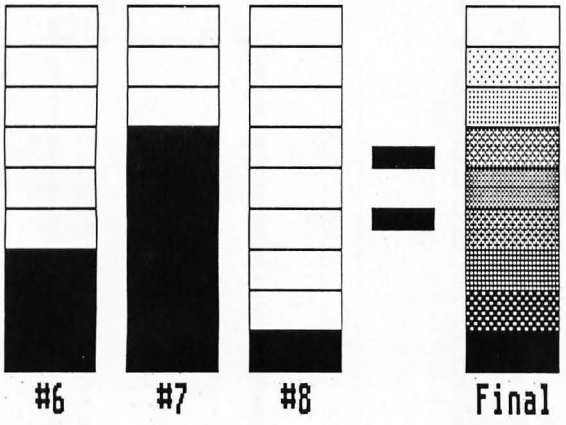



**Figure 5B**






what's the point of all this if it blinks at you?" you might say. Well, I use my camera to take photos of the monitor's screen to help in illustrating articles. You see them in this article, for example. If I know that a certain photo is to appear on a black-and-white page, I prefer to shoot it off a monochrome screen. Unlike a color monitor, scan lines do not as readily appear in monochrome. There are just too many of them to be noticeable. The normal method of rendering grays in monochrome by using fill patterns is, however, very noticeable.

Remember that I said I use a camera to take screen shots? Well, photographic film exhibits an extremely high degree of "persistence" as an input device. This fact has been exhibited for decades in the field of astronomy where they direct light coming through a telescope onto a sheet of photographic film for hours on end, allowing the exposure to build up. That $^{1}/_{10}$ or $^{3}/_{10}$ of a second flicker rate will not be noticeable when the camera's shutter is left open for one or two seconds!

## And now for something more BASIC

The GFA BASIC 3.0 routines presented here will allow you to plot, draw lines, shapes and text using multiframe gray scaling. For the most part, these routines are merely substituted for the standard BASIC graphics commands like PLOT, LINE, DRAW TO, TEXT, CIRCLE, BOX, etc. As an example, let's say that we want to draw a circle in the middle of our monochrome screen with a radius of 50 and a gray-scale value of 4. Normally we would use the command:

```
CIRCLE 320,200,50/PX
```

to accomplish this. We, however, want to draw in several different frames to get the gray effect, so we call the GFA BASIC procedure *@gcircle* as follows:

```
gcolor&=4 @gcircle(320,200,50)/PX
```

which first sets the gray color value to 4 and then plots the circle in all the frames. This assumes that we've already set aside or dimensioned the different areas of RAM that will hold the frames and initialized variables that point to them. All of the procedures and global variables used by these gray-scale routines start with the letter *g* for gray. There is no real reason for this; it just makes it easier to debug and document the code.

## Turning gray

Now, lets get down to the "meat and potatoes" of the program and examine how the Mono-Gray procedures work. The first block of code to get executed (or else none of this will work) is:

```
@ginit(planes&)
```

which happens to be one of the longest pieces of code here and with good reason. It lays the groundwork for the rest of the Mono-Gray code to work in by reserving screen RAM, initializing pointers and variables and clearing the screen. It is called as shown above by substituting the number of planes you want to use as the parameter. For example, if you want to be able to display eight different brightness levels numbered [0..7], this would require seven planes. Four brightness levels numbered as [0..3] would require three planes. The number of planes specified must be an integer greater than one. Specifying too large a value (each plane using up 32,000 bytes) would normally produce an "out of memory" error, but I've added some code to check for this instead. Please note that this procedure can be called only once when you run your program, otherwise it would try to dimension arrays twice had a little bit of error-checking code not been added.

Let me also explain the variables that are found in *@ginit* so you'll have a better idea as to what is going on, but more importantly, so you won't use them in your own code.

*gmax&*—This holds the value passed to the *@ginit* procedure and is used by just about every Mono-Gray routine. There is no reason that your code should change this, but it can be read to determine the number of planes used. Most of the variables are like this so I'll only tell you which ones are safe to use and why. Also, most of these variables you'll see here are two-byte word-length variables found in Version 3.0 of GFA BASIC. They can be converted to 2.0 style integers if need be. I prefer 3.0 and use it exclusively.

*gptr%( )*—An array of long integers that point to the beginning of each frame of screen RAM. They are used by all those XBIOS commands to change drawing and/or viewing frames.

*gram|( )*—A large array of type BYTE to hold all the screen frame data.

*glogbase%*—Holds the address of the original area of screen RAM so we can put it back when we're done. Used by the *@gexit* procedure.

*gcolor&*—This one you do want to include in your code, for it allows you to specify the drawing color. Just set this variable to the selected shade of gray before calling the drawing routines for that color. Initialized here to *gmax&*.

*glastx&*—Set to the X-coordinate endpoint by the *@gplot*, *@gline* and *@gdrawto* routines.

*glasty&*—Y-coordinate match for *glastx&*.

Our next routine is called after the drawing is complete. Its job is to cycle through the graphics frames in tune with the vertical blank routines (so there are no visible glitches) to display the total picture. It is called by the command:

```
@gshow
```

It requires no parameters since it gets all its data from variables that were set up in *@ginit*. The code is written to exit when either mouse button is pressed, but can be easily modified to trigger on keyboard input or via a built-in time delay. Upon exiting, you will be once again looking at the first Mono-Gray screen.

We even have a procedure to erase all of the Mono-Gray drawing screens, which also doesn't require parameters. Since it is similar to the GFA BASIC command CLS, we give it a similar name and call it as follows:

```
@gcls
```

Before exiting our BASIC program, we need to put the screen RAM back to where it was before we ran the *@ginit* procedure. This is accomplished by the routine:

```
@gexit
```

## The PLOT thickens

Now we get down to the nitty-gritty and give the procedures that do the actual writing to the screen (screens?). We'll start off with our routine to plot dots, which is called as follows:

```
@gplot(x&,y&)
```

This is just like the GFA BASIC statement *PLOT x&, y&* with the values of the variables, constants or numeric expressions being passed following the same rules. The coordinate values are best limited to [0..639] for X and [0..399] for Y. This holds true for the other gray-scale output routines.

The plotting color (or grayness in this case) used in all these graphics routines is determined by the value of the variable *gcolor&*. To select a new shade of gray, simply equate this variable to the new value like this:

```
gcolor&=(5+x)/2
```

Note that the value used must be a positive integer, less than or equal

to the value of *gmax&* as defined in the routine *@ginit()*.

The next two commands are related in that they both draw lines to the screen and look like this:

```
@gline(x1&,y1&,x2&,y2&)
@gdrawto(x&,y&)
```

The first procedure, *@gline*, is equivalent to the GFA BASIC statement *LINE x1,y1,x2,y2* and requires the starting and ending X and Y coordinates to work. The *@gdrawto* procedure is akin to the command *DRAW TO x,y* in that it also draws lines but takes the starting point from the last coordinate pair used by the most recent call to *@gplot*, *@gline* or *@gdrawto*. You can draw connected lines by calling the *@gdrawto* routine a number of times. Note that you can use the BASIC statement DEFLINE to change things such as line style, line width or specifying round, square or arrow-shaped line ends. DEFLINE can also be used with the upcoming code to change the characteristics of drawn boxes and circles.

```
@gbox(x1&,y1&,x2&,y2&)
@gpbox(x1&,y1&,x2&,y2&)
@grbox(x1&,y1&,x2&,y2&)
@gprbox(x1&,y1&,x2&,y2&)/PX
```

The above are the four routines to generate boxes on the screen. In the order that they appear above, they generate boxes, filled boxes, boxes with rounded corners and filled boxes with rounded corners.

The DEFLINE statement, as explained with the plotting and line drawing routines, can be used to change the characteristics of the outer box. The filled box commands (as well as the filled circle and ellipse commands to follow) may also use the DEFFILL statement to select a specific fill pattern.

```
@gcircle(x&,y&,radius&)
@gpcircle(x&,y&,radius&)
```

The above call draws circles and filled circles respectively. Simply specify the X and Y coordinates for the center of the circle as well as its radius and it will be drawn in the color specified by the variable *gcolor&*.

```
@gellipse(x&,y&,rx&,ry&)
@gpellipse(x&,y&,rx&,ry&)
```

These calls make available to us the ellipse drawing routines found in GFA BASIC. The parameters that substitute for *rx&* and *ry&* determine the radius from center in the horizontal and vertical directions. I guess there is no easy way to draw an ellipse that is set at a 45-degree angle!

We have modified the *@gcircle* and *@gpcircle* routines to allow them to handle drawing parts of a full circle by specifying the starting and ending angles in 1/10 of a degree [0..3600]. These procedures are:

```
@garc(x&,y&,radius&,angle1&,angle2&)
@gpie(x&,y&,radius&,angle1&,angle2&)
```

They are identical to the code that generates circles and, correspondingly, filled circles with the addition of the angle parameters.

Finally, the same thing has been done to produce partial ellipses and filled ellipses with:

```
@gearc(x&,y&,rx&,ry&,angle1&,angle2&)
@gepie(x&,y&,rx&,ry&,angle1&,angle2&)
```

Text may be presented on the screen in gray scale by the use of the Mono-Gray procedure:

```
@gtext(x&,y&,string$)
```

This procedure takes the same parameters as the GFA TEXT statement to specify the starting X and Y coordinates, as well as the string to write. The DEFTEXT statement can be used to change the characteristics of the

drawn text. Bold, light, italics, underlined, outlined or any combination of the above text attributes can be selected with DEFTEXT, as well as specifying the angle and size of text.

To determine the shade of gray of a particular point on the screen, use the following function:

```
gray&=@gpoint(x&,y&)
```

This is essentially the opposite of the plotting routines in that it adds up the values found by the BASIC function *POINT(x,y)* for the same coordinates on all the different planes and returns the total.

## To FILL or not to FILL

I had decided to leave out the implementation of the graphic statement FILL that is found in GFA BASIC, since I was sure that I couldn't guarantee it would work for all cases. For example, say you wanted to draw a circle in Shade 1 but fill it with Shade 2. Shade 2 requires writing to two separate graphics planes, whereas the circle drawn in Shade 1 appears in only one plane. The fill operation would work properly in the first plane, filling in the circle, but the circle would not be there when it came to fill the second plane. The fill operation would then overflow the area. That's nasty, so I'd decided not to do it.

Then I decided to, in fact, do it— provided that those who tried to implement the routine:

```
@gfill(x&,y&)
```

If you implement the routine, you must understand the aforementioned limitations. It can only be guaranteed to work if the fill color (set by the variable *gcolor&*) matches the color of the closed figure you wish to fill, or if the closed figure's color is the maximum allowed by the resolution specified in *@ginit()*, which is also the value of the Mono-Gray variable *gcolor&*. It will also always work if the fill color is set to 1 (the lowest brightness before total black), and the color of the object to be filled is at least one. There are most certainly other color combinations that will produce the desired results, but these depend on the gray resolution used. If someone can come up with a mathematical formula that can test for overflows produced by the *@gfill* procedure, I'd certainly like to see it.

## Listings and such

The GFA BASIC code to do all of the above can be found in Listing 1. This listing also contains a short demo program to exhibit some of the features of Mono-Gray. Listings 2 and 3 can replace the demonstration code in Listing 1 to show off other features of the program. If you do not wish to type in these programs, you can find them on this month's disk version or as downloadable files in the ST-LOG SIG on DELPHI.

Next month, I'll present more BASIC code to convert and display low resolution *DEGAS* and *Tiny* files using the techniques presented here. In the meantime, I'll be working on a machine language desk accessory for a future issue that will display these, as well as *Neochrome* and *Spectrum 512* files. Until then, may your blue skies be gray! ▪

**There are most certainly other color combinations that will produce desired results, but these depend on the gray resolution used. If someone can come up with a mathematical formula that can test for overflows produced by the *@gfill* procedure, I'd certainly like to see it.**

*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**

Monochrome-Gray

```
' MONO-Gray Graphic Routines
' by Charles Bachand for ST-Log magazine
' Copyright 1989 by ST-Log
' ------------------------------------------------------------------------
'
' *** Note:  Your MONO-Gray program goes here! ***
'
' Demo program to draw random lines in three brightness levels.
'
@ginit(3)                 ! three shades of gray
DEFLINE 1,5,1,1           ! 5 pixel wide lines with arrow ends
FOR gcolor&=1 TO 3        ! rotate through gray values
  FOR j&=1 TO 15          ! draw 15 random lines
    @gline(RAND(640),RAND(400),RAND(640),RAND(400))
  NEXT j&                 ! draw next arrow
NEXT gcolor&              ! change gray color
DEFTEXT ,0                ! select normal text
@gtext(20,20,"Press either mouse button to exit.")
@gshow                    ! rotate planes -> exit on mouse button
@gexit                    ! clean up
END
'
' ------------------------------------------------------------------------
'
' ---------------------------------
' MONO-Gray program startup routine
' ---------------------------------
'
PROCEDURE ginit(planes&)
  LOCAL index&,g&
  IF XBIOS(4)<>2
    ALERT 3,"MONO-Gray is for use with|Monochrome monitors only!",1,"OK",g&
    END
  ENDIF
  IF gmax&<>0
    @gexit
    ALERT 3,"Procedure GINIT() can not|be called more than once!",1,"OK",g&
    END
  ENDIF
  IF FRE(0)<planes&*32004+255
    g&=INT((FRE(0)-255)/32004) ! maximum planes possible with present memory
    ALERT 3,"Too many MONO-Gray|planes.  Maximum is "+STR$(g&)+".",1,"OK",g&
    END
  ENDIF
  gmax&=planes& ! used by other routines to determine number of planes
  DIM gptr%(gmax&) ! pointers to screen RAM areas
  DIM gram|(32000*gmax&+255) ! RAM area for gmax& screens
  ARRAYFILL gram|(),0
  gptr%(1)=VARPTR(gram|(0))+255 AND &HFFFF00 ! find first screen address
  FOR index&=2 TO gmax&
    gptr%(index&)=gptr%(index&-1)+32000 ! other screen addresses
  NEXT index&
  glogbase%=XBIOS(2) ! old screen RAM address used by GEM
  VOID XBIOS(5,L:-1,L:gptr%(1),-1) ! display first screen (gray level 1)
  gcolor&=gmax& ! initial drawing color of range [0..gmax&]
  glastx&=0 ! used by @gdrawto command
  glasty&=0
RETURN
'
' ---------------------
' MONO-Gray procedures
' ---------------------
'
' Display gray scale graphics
'
PROCEDURE gshow
  LOCAL index&
  REPEAT
    FOR index&=1 TO gmax&
      VSYNC
      VOID XBIOS(5,L:-1,L:gptr%(index&),-1)
    NEXT index&
  UNTIL MOUSEK
  VOID XBIOS(5,L:-1,L:gptr%(1),-1)
RETURN
'
' Clear screens
'
PROCEDURE gcls
  ARRAYFILL gram|(),0
RETURN
'
' Exit MONO-Gray routines
'
```

```
PROCEDURE gexit
  VOID XBIOS(5,L:glogbase%,L:glogbase%,-1)
RETURN
'
' Plot in gray
'
PROCEDURE gplot(gx&,gy&)
  @gcode(1)
  glastx&=gx&
  glasty&=gy&
RETURN
'
' Draw lines in gray
'
PROCEDURE gline(gx1&,gy1&,gx2&,gy2&)
  @gcode(2)
  glastx&=gx2&
  glasty&=gy2&
RETURN
'
' Gray DRAWTO procedure
'
PROCEDURE gdrawto(gx&,gy&)
  @gcode(3)
  glastx&=gx&
  glasty&=gy&
RETURN
'
' Draw box in gray
'
PROCEDURE gbox(gx1&,gy1&,gx2&,gy2&)
  @gcode(4)
RETURN
'
' Draw filled box in gray
'
PROCEDURE gpbox(gx1&,gy1&,gx2&,gy2&)
  @gcode(5)
RETURN
'
' Draw rounded box in gray
'
PROCEDURE grbox(gx1&,gy1&,gx2&,gy2&)
  @gcode(6)
RETURN
'
' Draw filled rounded box in gray
'
PROCEDURE gprbox(gx1&,gy1&,gx2&,gy2&)
  @gcode(7)
RETURN
'
' Draw circle in gray
'
PROCEDURE gcircle(gx&,gy&,gr&)
  @gcode(8)
RETURN
'
' Draw filled circle in gray
'
PROCEDURE gpcircle(gx&,gy&,gr&)
  @gcode(9)
RETURN
'
' Draw ellipse in gray
'
PROCEDURE gellipse(gx&,gy&,grx&,gry&)
  @gcode(10)
RETURN
'
' Draw filled ellipse in gray
'
PROCEDURE gpellipse(gx&,gy&,grx&,gry&)
  @gcode(11)
RETURN
'
' Draw circlular arc in gray
'
PROCEDURE garc(gx&,gy&,gr&,gw1&,gw2&)
  @gcode(12)
RETURN
'
' Draw filled circlular arc in gray
'
PROCEDURE gpie(gx&,gy&,gr&,gw1&,gw2&)
  @gcode(13)
RETURN
'
' Draw elliptical arc in gray
'
PROCEDURE gearc(gx&,gy&,grx&,gry&,gw1&,gw2&)
  @gcode(14)
RETURN
'
' Draw filled elliptical arc in gray
'
PROCEDURE gepie(gx&,gy&,grx&,gry&,gw1&,gw2&)
  @gcode(15)
RETURN
'
' Draw text in gray
'
PROCEDURE gtext(gx&,gy&,gstring$)
  @gcode(16)
RETURN
'
' Determine gray color [0..gmax&] of selected point
'
FUNCTION gpoint(gx&,gy&)
  LOCAL index&,val&
  val&=0
  FOR index&=1 TO gmax&
    VOID XBIOS(5,L:gptr%(index&),L:-1,-1)
    IF POINT(gx&,gy&)
      INC val&
    ENDIF
  NEXT index&
  RETURN val&
ENDFUNC
'
' Fill in gray
'
PROCEDURE gfill(gx&,gy&)
  @gcode(17)
RETURN
'
' ------------------------------------------------
' Common code called from other MONO-Gray entry points
' ------------------------------------------------
'
PROCEDURE gcode(code&)
  LOCAL index&,gcounter&,tempc&
  gcounter&=0
  FOR index&=1 TO gmax&
    SUB gcounter&,gcolor&
    IF gcounter&<0
      ADD gcounter&,gmax&
      tempc&=1
    ELSE
      tempc&=0
    ENDIF
    VOID XBIOS(5,L:gptr%(index&),L:-1,-1)
    SELECT code&
    CASE 16
      DEFTEXT tempc&
    CASE 5,7,9,11,13,15,17
      DEFFILL tempc&
    DEFAULT
      COLOR tempc&
    ENDSELECT
    SELECT code&
    CASE 1
      PLOT gx&.gy&
    CASE 2
      LINE gx1&,gy1&,gx2&,gy2&
    CASE 3
      LINE glastx&,glasty&,gx&,gy&
    CASE 4
      BOX gx1&,gy1&,gx2&,gy2&
    CASE 5
      PBOX gx1&,gy1&,gx2&,gy2&
    CASE 6
      RBOX gx1&,gy1&,gx2&,gy2&
    CASE 7
      PRBOX gx1&,gy1&,gx2&,gy2&
    CASE 8
      CIRCLE gx&,gy&,gr&
    CASE 9
      PCIRCLE gx&,gy&,gr&
    CASE 10
```

```
      ELLIPSE gx&,gy&,grx&,gry&
    CASE 11
      PELLIPSE gx&,gy&,grx&,gry&
    CASE 12
      CIRCLE gx&,gy&,gr&,gw1&,gw2&
    CASE 13
      PCIRCLE gx&,gy&,gr&,gw1&,gw2&
    CASE 14
      ELLIPSE gx&,gy&,grx&,gry&,gw1&,gw2&
    CASE 15
      PELLIPSE gx&,gy&,grx&,gry&,gw1&,gw2&
    CASE 16
      TEXT gx&,gy&,gstring$
    CASE 17
      FILL gx&,gy&
    ENDSELECT
  NEXT index&
RETURN
```

**MONO-GRAY**
***Listing 2:***
**GFA BASIC**

```
'
' Demo program to draw Moire pattern in two brightness levels.
'
@ginit(2)                 ! two shades of gray
FOR i&=0 TO 639 STEP 2
  INC color&
  gcolor&=color& MOD gmax&+1
  @gline(0,i&/2,i&,399)
  @gline(639,i&/2,639-i&,399)
  @gline(0,399-i&/2,i&,0)
  @gline(639,399-i&/2,639-i&,0)
NEXT i&
gcolor&=gmax&
@gtext(184,208,"Press either mouse button to exit.")
@gshow                    ! rotate planes -> exit on mouse button
@gexit                    ! clean up
END
```

**MONO-GRAY**
***Listing 3:***
**GFA BASIC**

```
'
' Display gray text examples.
'
SETCOLOR 0,0  ! inverse video
@ginit(3)     ! set up three frames
gcolor&=3     ! set gray shade to white
@gtext(50,20,"Demo of text in 3 shades of gray.")
@gtext(50,200,"Press either mouse button to EXIT.")
FOR i&=0 TO 2
  gcolor&=i&+1
  DEFTEXT ,0  ! set normal text
  @gtext(10+200*i&,50,"Normal mode")
  DEFTEXT ,1  ! set bold text
  @gtext(10+200*i&,66,"Bold mode")
  DEFTEXT ,2  ! set light text
  @gtext(10+200*i&,82,"Light mode")
  DEFTEXT ,4  ! set italic text
  @gtext(10+200*i&,98,"Italic mode")
  DEFTEXT ,8  ! set underlined text
  @gtext(10+200*i&,114,"Underlined mode")
  DEFTEXT ,16 ! set outlined text
  @gtext(10+200*i&,130,"Outlined mode")
NEXT i&
@gshow
@gexit
END
```

# Monochrome-Gray

# FONT ID EDITOR

by Charles F. Johnson

Maurice Molyneaux's *Step 1* column this month is about GDOS, which stands for graphic device operating system, (although people have been known to call it other, less publishable names, as well). One of the purposes of GDOS is to let applications use multiple fonts for both screen and printer output. The scheme that GDOS uses to manage the fonts, however, is anything but "user friendly," and it puts a lot of the burden of font management squarely on the users' shoulders. Since Maurice has already explained the whole process very well, I'll refer you to his column for the down and dirty details.

## THE PROBLEMS

As I was reading over a rough draft of Maurice's column, it occurred to me that there was really no way to easily change the ID number of a GDOS font. And as the column says, there are some occasions where you may need to do just that, since GDOS uses the font ID numbers to tell which fonts belong to the same "family" (e.g., Swiss, Dutch, typewriter, etc.). A GDOS font family may consist of several files, each representing a different point size. The only way GDOS knows which fonts belong to the same group is by their font ID numbers. The filenames are meaningless. All the point sizes of a font must have the same ID number or else GDOS cannot group them together.

With more public domain and commercial GDOS fonts appearing all the time, it's very possible to end up with two different fonts that have the same ID number. When these fonts are listed in the ASSIGN.SYS file, GDOS is unable to tell them apart, and you've got trouble. The problem is that the font IDs are contained *within* the font files, and the only way to change them is to use a program like Neocept's *Fontz!* or Brad Christie's public domain GEMFED.PRG. (If you're adventurous, you could also use a disk sector editor. Just remember that all values in GDOS font files are stored in 8088 "low/high" format.)

There are other problems related to font ID numbers too. Some GDOS applications expect the fonts to be loaded with their font IDs in ascending order. This is usually fine as long as you use the fonts and ASSIGN.SYS file supplied with the program. But if you ever need to add or delete fonts from the list, watch out! And to make things even more confusing, some programs cannot recognize fonts with ID numbers higher than 255.

## THE SOLUTION

*Font ID Editor* is a short, simple and easy-to-use utility program that lets you view and change the ID number of a GDOS font file. To use it, type in the ST BASIC listing accompanying this article (and check your typing with *ST-Check* on page 48), save the ST BASIC file, and then run it. (You don't need to type in the assembly language source-code listing. It is included for those readers interested in programming.) The BASIC program will create a file called ID__EDIT.PRG. You can now run *Font ID Editor* by double-clicking this file from the ST desktop.

*Font ID Editor* uses the GEM file selector to let you locate and choose a GDOS font to examine. When you select the file, the program displays its full pathname, the font name, point size and current ID number, and asks you to type in a new ID number. At this point, you can send the font information to your printer by typing the letter *P* (upper or lowercase) and hitting Return. This makes it simple to keep lists of fonts and their ID numbers—which is a very good idea if you have a lot of fonts.

If you wish to change the font ID, type in a number and hit Return, and *Font ID Editor* will write the new ID to the chosen file. Note that you can use any number from two to 32,767 for your font IDs, but there are a few conventions and restrictions. Again, be sure to read *Step 1* for some tips on the best way to number your fonts. If you hit Return without typing anything, the ID number will not be changed. After each font, you will be asked if you wish to examine another font. This way you can quickly change the ID numbers of an entire font family.

I hope *Font ID Editor* makes it a little bit easier for you to find your way through the GDOS jungle! ■ *(to page 72)*

# INTRODUCING

# VideoGames & Computer Entertainment™

## 12 ALL COLOR ISSUES ONLY $19.95

**Save over $15 off the cover price!**


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

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

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

☐ Payment Enclosed — Charge My ☐ VISA ☐ MC

_____ EXP_____

SIGNATURE_____

NAME_____

ADDRESS_____

CITY_____ STATE_____ ZIP_____

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

Your first issue will arrive in 6 to 8 weeks.
WATCH FOR IT!!
Offer expires June 28, 1989

CJQWA

# 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 |

| GRAPHICS | |
|---|---|
| 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 1000cl | 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*

| GAMES | |
|---|---|
| Tanglewood | 27.95 |
| Test Drive | 35.00 |
| Chessmastr2000 | 32.95 |
| Starglider II *NEW* | 35.00 |
| Hunt for Red Oct | 35.00 |
| TyphoonThompson | 27.95 |
| Aliants | 24.95 |
| Alien Fire | 35.00 |
| Santa Paravia | 21.00 |
| Lurking Horror | 27.95 |
| Star Fleet 1 | 39.95 |
| Empire | 39.95 |
| Liesure Suit Larry | 27.95 |
| Gridiron | 35.00 |
| Dungeon Master | 27.95 |
| Flight Simulator | 35.00 |
| Trailblazer | 27.95 |
| ····SPECIALS···· | |
| Jewel of Darknss | 19.95 |
| Silicon Dreams | 19.95 |
| Cardiac Arrest | 48.95 |

| The Best European Games | |
|---|---|
| I Ball (neat,fast) | 19.95 |
| RanaRama(d&d) | 29.95 |
| Warlock'sQuest | 29.95 |
| The Flintstones | 29.95 |
| Trivial Pursuit | 29.95 |
| The Enforcer | 19.95 |
| Seconds Out | 29.95 |
| Scruples(board) | 29.95 |
| Livingstone | 29.95 |
| Battle Ships | 29.95 |
| Outrun(fast cars) | 29.95 |
| Crazy Cars | 29.95 |
| Tetris (from USSR) | 29.95 |
| Screaming Wings | 29.95 |
| Spitfire | 29.95 |
| Blue War | 29.95 |
| Star Quake | 29.95 |
| Enduro Racer | 29.95 |
| BMX Simulator | 29.95 |
| Arkanoids | 29.95 |
| 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 #106 ON READER SERVICE CARD.

# STEP 1: Understanding Atari GDOS Part 1

I have to admit. I've put this topic off for as long as I thought I possibly could. Not because I didn't want to write about it, but, to be perfectly honest, because I didn't feel I understood the subject enough to do it justice. In the past few months I've become much better acquainted with GDOS, so now *Step 1* returns to its roots, explaining the mysteries of the ST, as we *finally* discuss Atari's infamous GDOS.

## What the heck is it?

GDOS is a part of GEM that was, for various reasons, not included in the ST operating system (OS). It is available now, as a small program that is run from an AUTO folder at boot-up that hooks into GEM and acts as if it were part of the OS. GDOS is an acronym for "graphics device operating system," which sounds more impressive than it really is. What GDOS does is load a set of specified graphics device drivers and fonts and allow metafiles, graphics and fonts to be output to a device at its maximum possible resolution.

Wait a second. We're already getting into too much terminology that most of you probably don't understand. Allow me to define a few key terms we'll be using.

*Device drivers:* These are special programs that take the information provided by a GDOS application and translate it to a form that can be used by a particular output device (like a printer or a monitor). Without such drivers, the information coming from the application would be meaningless to the output device.

*GDOS application:* A program which uses GDOS and its device drivers, fonts, etc.

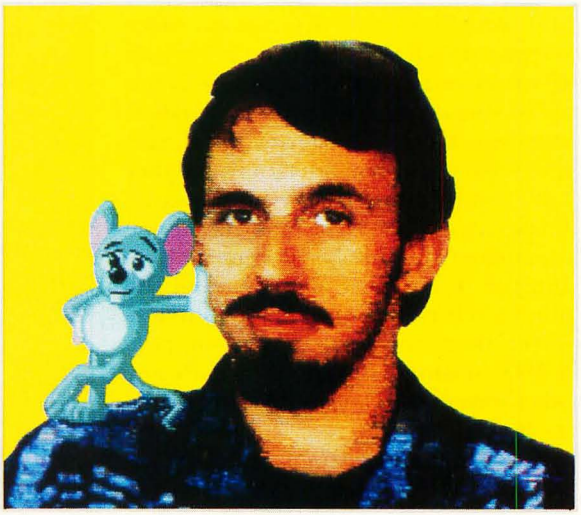
*GDOS fonts:* A font is a complete selection of type in one style, and with GDOS it is also one *size*. Thus, if you wanted to use a Gothic font in four different point sizes, GDOS would have to load four different font files, each one a font of the appropriate size. If you wanted to print the font as well, you'd also have to load four printer fonts equivalent to the screen fonts! Currently GDOS doesn't support rescalable, vector plotted fonts, so a font doesn't look very good when rescaled. ("Vector plotted" means the symbols in the font are defined by a series of points, lines and arcs, and scaling them up and down involves nothing more than increasing or decreasing the distance between points and the size of arcs.)

*Metafile:* A metafile is a (supposedly) standardized file that can be used by many different GDOS applications. It is an image stored in a resolution-independent (meaning it can be used on any output device, regardless of its maximum resolution) .GEM file, which stores all information about the image, such as commands on how the image was drawn (the actual commands for drawing lines, ellipses, etc., in addition to storing the font information in a similar way). Using this information, a .GEM file-compatible program should be able to recreate the image and output it to a printer and/or monitor. In this manner, you should be able to create a metafile with *Easy Draw* (Migraph) and use it in *Timeworks Desktop Publisher ST* (Timeworks). While this is what metafiles are *supposed* to do, sometimes they don't. Unfortunately, .GEM files haven't been completely standardized, so metafiles from some programs can not be utilized by others.

Getting back to what GDOS does, the following are its main uses(please excuse a small amount of repetition):

—Loads multiple output device "drivers" to allow printers, monitors, etc., to display/output information passed down by GDOS applications. Resolutions of output devices can be up to a maximum of 32,727 x 32,727, including emulation of this resolution on a monitor

by Maurice Molyneaux



screen! This is a huge number of points. For example, a laser printer with 300 dpi (dots per inch) resolution would at maximum be able to output 2,250 x 3,300 on an 8.5 x 11-inch sheet of paper, which is under 1/10 of the maximum resolution allowed under GDOS.

—Allows a program to load and utilize multiple fonts in various point sizes that can be displayed on a monitor, or output to a printer, plotter or other output device.

—Allows the application to create .GEM metafiles that are useful for moving images and data from one GDOS application to another.

While not a lot of ST programs use GDOS now, the number is increasing slowly. If you buy any amount of applications software, sooner or later you are likely to pick up a GDOS application. Programs that use GDOS include *CAD-3D, DEGAS Elite, Easy Draw, Timeworks Desktop Publisher ST* and *WordUp.*

As to *why* a program would use GDOS, the aforementioned portability of metafiles is one reason, and the device-independent nature of the images created with it are another. Because GDOS applications usually build images using vector-plotting routines, the images can be rescaled endlessly with no loss in detail (if you make the image too small it's possible your output device won't be able to display enough detail, and the image may appear garbled; but it's still there in its entirety).

## Versions and releases

Over the past two years there have been several revisions of GDOS, all sporting a "version" number. In the very recent past, Atari has supplied revisions sporting a "release" number. Apparently the "version" ones were pre-release models, and the "release" ones are *finished* models. Most programs written to use an older version of GDOS will usually work with all newer versions, but the reverse is not always true (as some older GDOS versions were incomplete or buggier). If you have several programs that use GDOS and they came with different versions/releases, check to see which is the most recent one (the higher the number the newer it is) and use it. If one is a version and the other a release, use the release as that is newer than any version. (If you have GDOS Version 1.8, replace it with any other version you can, because apparently it's buggy.)

To see the version/release number, you will have to boot your ST with a GDOS in the AUTO folder. When run, GDOS will display a message like: *Atari GDOS Release 1.1 Installed.*

Although this is not the place to go into details on all it can do—since I'm talking about GDOS versions—I have to mention that there is a commercial replacement for GDOS available, which is so good that if you use GDOS regularly, you really should have it. It's called *G+Plus* and is distributed by CodeHead Software (more on it later).

## Order now!

As to where you *get* GDOS: It is usually supplied with the application program that uses it. It is not offered as a commercial product or in any other fashion.

As to *using* it: There's not much to it. All you have to do is put the GDOS.PRG program into the AUTO folder of your boot drive and that's it. But, GDOS alone won't do anything. You need to provide it with information as to which drivers to load, etc. This data is provided by a file by the name of. . .

## ASSIGN.SYS

The ASSIGN.SYS file is the main element required for GDOS to do anything at all. You could have ten device drivers and 50 fonts, but without one ASSIGN.SYS file, they would all be useless. ASSIGN.SYS is nothing more than an ASCII text file that specifies what GDOS is to use and where to find it (most GDOS applications will supply a sample ASSIGN.SYS file).

The ASSIGN.SYS file must be in the root directory of the boot drive. A typical one might look like this:

```
PATH=D:\FONTDRIV.SYS\  ; Driver & font path
;
01p SCREEN.SYS  ; Default screen driver
;
02p SCREEN.SYS  ; Low resolution screen driver
ATSS10.FNT      ; Low res screen fonts
ATSS18.FNT
ATTP10.FNT
ATTR18.FNT
;
03p SCREEN.SYS  ; Medium resolution driver
ATSS10CG.FNT    ; Medium res fonts
ATSS18CG.FNT
ATTR10CG.FNT
ATTR18CG.FNT
;
04p SCREEN.SYS  ;High resolution driver
ATSS10.FNT      ;High res fonts
ATSS18.FNT
ATTR10.FNT
ATTR18.FNT
;
21 FX80.SYS  ; Epson 9-pin dot-matrix printer
ATSS10EP.FNT ; Epson printer fonts
ATSS18EP.FNT
ATTR10EP.FNT
ATTR18EP.FNT
;
31r META.SYS  ; Metafile driver -- "resident"
```

If you have one or more GDOS applications, you might want to take a look at its ASSIGN.SYS file right now and compare it to the one here (you can either print it or just show it on the screen (double-click on ASSIGN.SYS and select the "show" button on the alert box that appears).

Let's go through it a line at a time. All text to the right of a semicolon is treated as a comment and ignored by GDOS.

*PATH = E: / FONTDRIV.SYS*

This tells GDOS from which directory path to load the device drivers and fonts. The slash following the last folder name isn't necessary, but neither does it do any harm.

*01p SCREEN.SYS*

This is the ST's "default" screen driver, one of four built-in screen drivers built into the machine. This driver is used when a GDOS application doesn't care about the screen resolution. The "01" tells GDOS that this is device driver number 01 (the built-in drivers are numbers 01 through 04). The "p" following the "01" is a flag used to tell GDOS that the driver is "permanent" (part of the ST OS) and that it does not have to be loaded. "SCREEN.SYS" is a place holder filename, as GDOS isn't really loading a driver. However, place holder or not, it *must* be present.

*02p SCREEN.SYS*

This is the built-in low-resolution screen driver. Immediately following it are the names of the six low-resolution font files loaded by this ASSIGN.SYS file.

*03p SCREEN.SYS*

The built-in medium-resolution screen driver, followed by the names of the

medium-resolution font files to be loaded.

*04p SCREEN.SYS*

The built-in high-resolution screen driver, followed by the names of the high-resolution font files to be loaded.

*21 FX80.SYS*

The number 21 tells GDOS that the filename following is a printer driver. "FX80.SYS" is the filename of the device driver for an Epson-compatible 9-pin dot-matrix printer (in my case a Star SG-10). Note that there is no "p" flag after the driver number as there was on the screen drivers. It also does not have an "r" flag as does the following item. If there is no flag present, GDOS will load only that device driver when an application opens/accesses that device. This could also be followed with a list of filenames of printer fonts to be loaded.

*31r META.SYS*

The "31" tells GDOS that this is a metafile driver. The "r" flag tells GDOS that the driver is to be a "resident" one, and that it should load it immediately and keep it in memory at all times. "META.SYS" is the filename of the driver proper.

There are cases where you may see a list of "fonts" following the metafile driver in an ASSIGN.SYS file. These are not fonts, in fact, but rather width-tables designed to keep character spacing correct in files to be used in different GDOS applications. (When you save a file as a metafile the fonts in it are saved as vector-plotted elements, thus becoming objects, rather than remaining as GDOS fonts.) Such "metafile fonts" are so rarely used that it's unlikely you will ever see one, and if you *do* happen to have some, you will probably discover that discarding and stripping them from your ASSIGN.SYS file will have little or no effect on your metafiles.

The ASSIGN.SYS file looks pretty simple, all in all. In many ways it is. However, creating or editing one can be a colossal pain in the bum. Before going on to how to create/edit one of these beasties, let's go into some more details on a few of the parameters seen in the above breakdown.

The first line in the file is the pathname for loading all drivers, fonts, etc. This really should be right at the beginning of the file, so there's no chance of GDOS looking for files in the wrong path. This line always begins with PATH= followed by a pathname of up to 64 characters.

Afterwards, all that is necessary is the list of device drivers and their associated fonts. In each case, what we have is a block beginning with a line identifying the driver, followed by a list of fonts. The driver identifier consists of three parts. First is a driver identification number. A list of these follows:

01-10 Screen drivers (01-04 are the built-in ST drivers)
11-20 Plotter drivers
21-30 Printer drivers
31-40 Metafile drivers
41-50 Camera drivers
51-60 Tablet drivers

(I have heard that some ASSIGN.SYS files also have a device 00 screen driver defined (as in "00 SCREEN.SYS") that is supposedly to help prevent incompatibilities between GDOS and some programs that seem to have trouble when GDOS is installed. Apparently while helping with some older programs, using device 00 can cause problems with other programs, so it's recommended that you not use it unless you absolutely have to.)

Please note that most GDOS applications allow you to use only one driver for an output device of a given category. The only program I know of that allows you to select between multiple drivers is Migraph's OUTPRINT (provided with *Easy Draw*). If you put multiple drivers under one category (such as printers), most programs will load the first one they encounter and ignore any others. But if you are using a program (like OUTPRINT) that allows you to select drivers, and you have more than one output device, you will have to give each a different ID number within the range for that type. For example, if you had both an FX-80 dot-matrix printer and a SLM804 laser printer, you would have to put both in the ASSIGN.SYS file, one before the other (probably in order of most use). So you might assign the FX-80 driver the ID number 21 and the SLM804 the ID 22.

After the driver number is the "load flag" (if any). These are the "r" (resident) and "p" (permanent) flags, as well as the absence of flags, discussed before.

Following the load flag is the filename of the driver. If it is one of the system's own screen drivers (01-04), then the "filename" must be SCREEN.SYS, otherwise the name must be the name of the file as it appears on disk.

Immediately below the driver identification line is a list of any and all fonts to be used by that driver.

## Creation

While most programs that use GDOS come complete with an ASSIGN.SYS file, that file may not always meet your needs or even be set up properly in the first place. If you want to install your fonts and drivers on a hard disk or on a drive other than the one set up in the default ASSIGN.SYS file, or if you want to delete or add fonts, you will have to either edit the file or create a new one.

To create or edit an ASSIGN.SYS file, you'll need only a program capable of saving an ASCII text file (no control codes). You can use *EMACS*, *1st Word* (word processor mode off), *Flash!*, *WordPerfect* (use the TEXTOUT function). Heck even *Cyber Control* will do it! If you're going to edit one, load it into the editor of your choice. If you want to create one, open a blank document and get ready to start typing. (For reference you may want to make a list of fonts and drivers before starting.)

The first thing you may want to do is title your file. Type a semicolon at the start of the topmost line in your file, and then type a name. For example, if this ASSIGN.SYS file is for *Easy Draw*, you may want to type something like the following as the first line in your file:

**The first line in the file is the pathname for loading all drivers, fonts, etc.**

```
; -- EASYDRAW.SYS --
```

This way you can view your file later and easily know what it is for (in the event that you forget or get some files confused). You may also want add comments after certain information, for reference purposes, or add lines to indicate the start of certain sections, as below:

```
21r FX80.SYS ; Epson 9-pin printer
```

or...

```
; -- LOW-RES SWISS FONTS --
```

Next, you need to set the path from which you'll be loading all drivers and fonts. I recommend putting these in a folder so they will be out of the way and not cluttering up often-accessed directories. If you are creating an ASSIGN.SYS file for a specific program, you may want to set a very specific path (especially if you have a hard disk). If you were creating one for *DEGAS Elite* and had that program on Drive D in a folder called ELITE, you might want to make a folder in that one called ELITE.SYS and put all drivers and fonts in it. Thus, your path line might look like this (remember, you *must* start the line with PATH = ):

```
PATH=D:\ELITE\ELITE.SYS\
```

If you are working off a floppy drive, you'll have to make sure you put the disk with the fonts and drivers in the same drive every time (easy if you only have one drive). If GDOS is also booted from floppy, your best bet would be to have the AUTO folder containing GDOS on the same floppy with the folder containing your fonts and drivers. If you have two floppy drives, you could put GDOS in the boot drive (A) and set the path for loading drivers and fonts to B. But I don't recommend this as it's easy to get confused and forget to put the proper disk in at boot-up.

Furthermore, if some drivers have no load flags set, the disk containing the driver and fonts will have to be in the drive specified by PATH = whenever you access the device in question, or the driver and fonts won't load. Furthermore, contrary to popular belief, GDOS does not load fonts on boot-up, but does so only when an application opens a device and requests the fonts (even if a driver is set as a "resident" one loaded at boot-up, the fonts aren't loaded until needed). Thus, the disk with the fonts and drivers would have to be present when this happens as well. So, to avoid confusion, put GDOS, ASSIGN.SYS, and the folder containing the fonts and drivers on one floppy and keep it in Drive A; then load the program you want to use from Drive B if you want (unless it is on the same disk as GDOS, etc., in which case this is both unnecessary and undesirable).

If you have a hard disk, but your system boots from Drive A (in other words, if it reads the AUTO folder from A), then you will have to keep GDOS in the AUTO folder of the floppy disk you use for booting your system. But you can set PATH = to the hard drive with no problems.

After setting up the PATH = line, all you have to do is type in the identification line for each driver, followed by a list of fonts to be used with it. In the case of screen drivers 01-04, which are in the ST's ROM, you will have no choice but to set the "p" flag (and there's no reason why you wouldn't want to). In all other cases you will have to determine whether you want the driver to be loaded at boot-up and stay resident (used by setting the "r" flag) or to load the driver only when the device is accessed (by setting no flags). The latter is probably the best way to go with printer and screen drivers. In this way all that GDOS loads at boot-up is the name of the driver and not the driver itself. So you don't lose RAM to the driver until it is needed and then it will be loaded.

As to listing the fonts under a device driver, apparently some GDOS applications have trouble finding fonts if you do not list the fonts in ascending order by font ID. (For information on finding out a font's ID number, see the end of this article.) All fonts of the same type have the same ID number, regardless of size. For example, all Helvetica fonts have the same ID number, regardless of whether they are 6 point or 30 point in size.

I have seen some users recommend creating a single all-purpose ASSIGN.SYS file for use with any and all programs you may have that use GDOS. I do not recommend this method, because creating one file to load all the necessary device drivers and fonts for several programs could be large, chew up a lot of memory when all fonts are loaded and run into potential problems with different font files having identical identification numbers and point sizes. (This is because GDOS ignores font names and your filenames for them; it goes by the font's internal ID number and point size.) It's unlikely you will run into this problem with an ASSIGN.SYS, which is for a single application, but you are more likely to run into it if you try to create an all-in-one file. (How to deal with multiple ASSIGN.SYS files will be discussed in the next issue.)

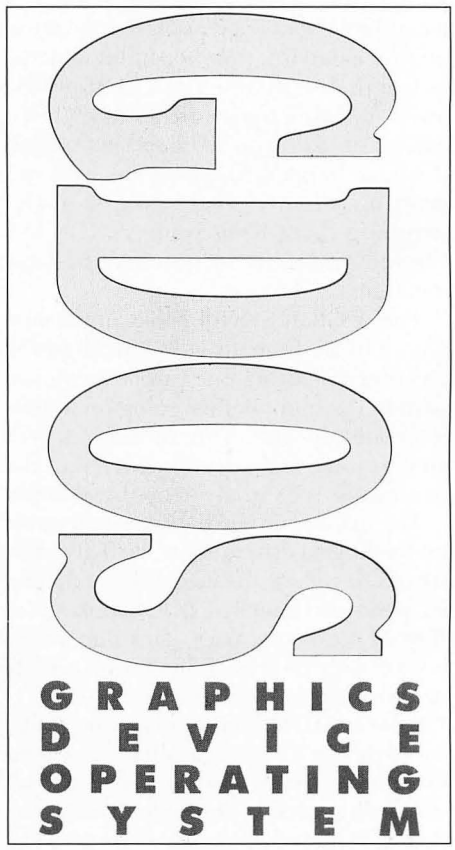
When you complete your ASSIGN.SYS file, you will want to save it (make sure it's ASCII output, unformatted—no tabs, control characters, etc.) to the root directory of the floppy disk or hard-drive partition that you will use as your boot directory. Once you've done this you'll have to reboot your system to see if your new assignments will work under GDOS. If not, you'll have to re-examine your ASSIGN.SYS file. Make sure everything is correct, and make sure your drivers and fonts are in the proper directory path. (It's easy to make a typo in either the ASSIGN.SYS file or in naming a folder or file.)

## Fonts

The way GDOS handles fonts is a subject about which there is probably more confusion than almost any other topic. Let's try to set the record straight. First and foremost, as I mentioned earlier, GDOS does *not* load the fonts defined in the ASSIGN.SYS file at boot-up, even if the driver they are installed under is a "resident" one. In fact, it seems GDOS flat out cannot load the fonts at boot-up, period! Only at such time as a GDOS application opens the device in question will GDOS attempt to load the fonts specified for it. If an error occurs when GDOS attempts to load one font (such as if it runs out of memory, or if it cannot find the font file), it will abort that load attempt and move on to the next font in the list.

Also, contrary to popular belief, GDOS fonts *are* unloaded when you exit an application, and any memory used by the fonts is then available for use by other programs.

GDOS fonts *can* be rescaled by an application, but because the fonts are not vector plotted, they don't usually rescale well and tend to come out looking bad.



Furthermore, neither GDOS nor the fonts it loads determine line spacing. The application program using the fonts does this. Sometimes, according to internal rules of its own, the system will rescale fonts and return them as valid point sizes to a running application. This rescaling is most often done to create large point sizes from smaller point sizes. For instance, the system may generate a 48-point font from a 24-point font, even though no 48-point font file exists. However, this "automatic scaling" is of limited use. The scaling method is actually a simple pixel-doubling algorithm that often gives the scaled font a blocky, "stair-stepped" appearance. If you can use actual fonts for the larger point sizes, the appearance will be much cleaner.

The filename of a font means nothing at all to GDOS. It reads size information and font ID numbers from values internal to the font file. The filenames are for the reference of the user only. About a year ago (as of this writing) Atari established a format for naming GDOS font files. A typical font filename might be ATTR06LB.FNT.

The first two characters identify the creator of the font. In this case, AT for Atari. (Atari suggests that all fonts for the ST begin with these two letters; but no one pays attention to that, and most users and companies will use something like NC for fonts originating at Neocept or MG for fonts originating at Migraph.)

The third and fourth characters identify the font style. In our example, it is TR, Atari's code for Times Roman. (Atari also uses SS for Sans Serif/Atari Swiss and TP for Typewriter style.) Other companies may have their own standards, in which case, the use of the first two characters to identify the origin of the font can be handy, for another company might use the same two letters for style that another does for a different typestyle.

The fifth and sixth characters contain the point size of the font. In our example, 06 stands for a 6-point font size. This number should ideally represent the size of the font as it appears on the output device, not the height of the font in dots, pixels, etc. As with the 06, use a zero to precede all single-digit point size values.

The seventh and eighth characters identify the output device for which the font is intended. In our example, LB identifies the font as for use with an Atari SMM804 dot-matrix printer. Other Atari-accepted identifiers for devices include: CG for color graphics, EP for Epson-compatible 9-pin dot-matrix printer, SP for Star NB-15 24-pin dot-matrix printer, LS for standard laser printer and MF for metafile.

The three-letter filename extender is always .FNT to identify the file as a font.

If you create your own fonts or obtain some whose names do not adhere to this standard, it may be to your benefit to name them along these lines in order to avoid confusion. If you are creating or importing fonts with style and device identifiers not already established, you may want to create your own standard list, such as GT for a Gothic font or CI for a Cannon Inkjet printer.

Before this standard was implemented, there was another font naming system employed. Since you may want to rename fonts using this old convention, I will describe it here. The first three characters of the filename were used to identify either the creator of the font (such as ATA for Atari) or the device it was intended for (such as EPS for an Epson 9-pin printer). The fourth character identified the resolution (such as using H for high-resolution). The fifth and sixth characters identified the typeface. (Atari's identifiers, such as TR for Times Roman, do not appear to have changed). The seventh and eighth characters were the point size. The extender, as above, was .FNT.

Quite a few GDOS fonts are floating around in the public domain. Some are commercially available. Furthermore, you can create your own fonts if you have the appropriate software (you can use Neocept's *Fontz!* or Brad Christie's excellent public-domain GEMFED (GEM Font Editor) for this purpose, as either will allow you to create both screen and printer fonts. *DEGAS Elite*'s font editing program can also be used, but only for screen fonts, and those can only be of limited detail (converting them to the correct format is a bit of a pain too).

When creating your own font, you must give the font a unique ID number. Make sure it isn't one already used by another font. (See addenda.) The ID can be any number from 0 to 32,767, though some GDOS applications seem unable to use fonts with an ID number over 255.

## Now it's time to say goodbye

Sheesh! At first I thought this article was going to run short! I underestimated. I'm not finished yet, so we'll have to conclude this discussion next issue, when we'll talk about how to keep more than one ASSIGN.SYS file and how to load the correct one for a given application, in addition to covering the problems spots of GDOS and giving a few examples of using GDOS in an application. Also, we'll discuss *G+Plus*, the GDOS replacement and why you might find it worth buying. Until then, *au revoir!*

*Addenda:* The author would like to thank Charles F. Johnson for his help in researching and editing this article. He also acknowledges Douglas Wheeler's excellent text *Everything You Ever Wanted to Know About GDOS (and More)*, which provided invaluable information and references.

In response to a question I asked him concerning how to find a GDOS font's ID number, Charles F. Johnson told me that the only way he knew of was to use a program like *Fontz!* or GEMDEF to load the font and check it. He then told me that

**When creating your own font, you must give the font a unique ID number.**

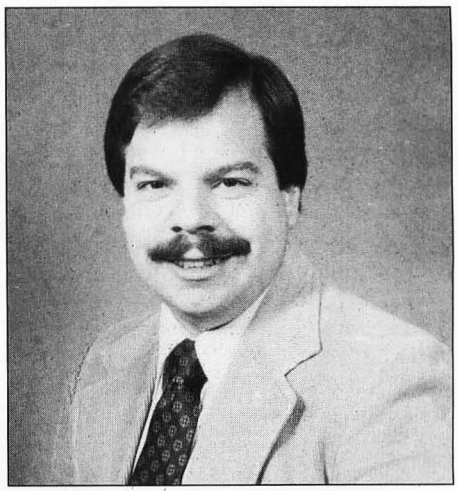
font and check it. He then told me that he could write a stand-alone utility to do that job; one that would also allow the user to change a font's ID number (in the event that two fonts had duplicate IDs). We discussed it; then he went off to write it. True to his word (and less than 12 hours later), it was finished and in my hands. You will find this useful utility on page 32 of this month's issue. ▪

---

*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.*

# ST USER

by Arthur Leyenberger



*Arthur Leyenberger is a human factors psychologist and freelance writer living in beautiful New Jersey.*

Have you looked around lately? I mean, *really* looked around. Have you seen them? You know, the... ah...eggheads. They're everywhere, and they seem to be multiplying faster than rabbits.

No, I don't mean those nerds in high school who used to carry three calculators around on their belts. Yeah, they're still around competing in chess matches, working in the MIS department of IBM or making a fortune programming video games. I mean the *Eggheads*, as in Egghead Discount Software stores!

"Now, wait a minute", you say. "I've seen their ads, driven past the stores. They don't sell ST software!" Right! They don't, but think about it for a minute. How many times have you gone to a computer store to purchase a specific program, and they either have never heard of it or didn't carry it? Or you weren't sure which program was just the right one for your particular needs and sought help from the computer dealer only to realize that Jeanne Dixon would have given better advice?

From the half dozen or so Egghead stores I have visited across the country, I can tell you that these salespeople know their stuff. Although each store I visited appeared to be cloned from some master mold, they all were well stocked, not only with the top ten IBM, Apple and Mac software titles, but many esoteric ones as well. In addition, every salesperson I talked to had intelligent answers to my PC and software-related questions. This is unheard of!

These stores are fantastic. They all offer a two-week, full refund or exchange if the program isn't exactly right. They guarantee to have the lowest price and will beat any lower advertised price. Just think what it would do for the ST market if stores like Egghead sold software for our machines. Think what it would mean to an ST user to be able to shop in a software store like this.

Egghead Software is a first-class operation. I assume they don't carry Atari software because there aren't enough STs out there. However, if Atari can get its act together, and the ST starts to really thrive, it would be foolish for Egghead to ignore the ST. We can only hope.

## High resolution

I know it's a little late—like maybe four months—but I want to make a New Year's resolution. I don't usually do this because it often doesn't last, but I think the time is right. For 1989, at least for the rest of the year, I resolve not to pick on Atari so much. Let me explain.

It's no secret that Atari has let the U.S. ST market dwindle away over the last couple of years. This is due to a number of causes both within and outside of Atari's control. Beyond Atari's immediate control, the shortage of DRAM (Dynamic Random Access Memory) chips has limited the production of ST computers. As a result, Atari has had to decide where it would sell the computers it was able to manufacture. Atari chose to concentrate on the European market where it has become one of the leading computer makers.

Atari decided to maintain its leadership position in the overseas market at the expense of the U.S. market. In addition, with few computers to sell in America, Atari decided not to advertise in the U.S., reasoning that advertising products that were virtually unavailable would only lead consumers to buy competing brands.

Now that the DRAM shortage is easing up, and Atari has stated that it has negotiated a deal for a plentiful supply of chips, there is no reason why Atari can't begin to make up the lost ground in the U.S. marketplace. Further, Atari has indi-

cated they *will* start advertising in the U.S. again, and perhaps by the time you read this, you will have seen the results. Ads are promised in noncomputing print media, such as the *New York Times*, *Wall Street Journal*, *Time* and other popular magazines. Even TV ads have been promised.

So. I'll take a chance on Atari (again). I'll believe them when they say they are serious about the U.S. market. I'll look forward to new products like the ST Laptop and the TT (68030) machine and others, all introduced at the appropriate time. I'll watch and hope that developers return to the fold. I'll keep on rooting for the Fuji. And most of all, I'll not pick on them any more for past errors.

Fair enough?

## Home automation

While perusing the contents of a magazine rack of a local bookstore, I happened upon a magazine I had not seen before. *Electronic House: The Journal of Home Automation* looked intriguing, so I bought it and took it home to read at my leisure. Interesting stuff.

Home automation deals with all types of home control systems—security devices, smart appliances, automatic heating, air-conditioning and watering systems. In one way or another, these devices use either dedicated microcomputers or personal computers to automate the functioning of household systems.

For example, I have mentioned one product in previous buyer's guides: The *X-10 System* from X-10 America, Inc. This is a product used to remotely turn AC devices on and off, either directly or from a preprogrammed timer. Assorted modules control such things as lights, appliances, thermostats and other electrical devices by means of commands sent through the AC wiring in your house. A computer peripheral called the *X-10 Powerhouse*, attaches to the RS-232 port of your ST and can be programmed to control these various AC devices.

The *X-10 Powerhouse* unit is self-powered and automatically controls the *X-10 System* modules. Once programmed, it can be disconnected from the computer, thus freeing the serial port for other uses. MichTron sells the *X-10 Powerhouse* for $25. Lamp and appliance modules sell for $10 to $15 and are available from Sears, Radio Shack, mail order and electronics specialty stores. In addition, replacement wall switches and outlets that look and work like ordinary fixtures are available for under $15.

The software that runs the *X-10 Powerhouse* is called *Echo* and is also published by MichTron. This $40 program was created especially for the *Powerhouse*/ST computer combination and allows you to use the X-10 system to change the status of up to 256 electrical devices at up to 128 different times of the week. *Echo* is a GEM application, so the mouse is used to enter all inputs and instructions and currently active desktop accessories are always available.

It appears that the home control field is currently at a similar stage to where the personal computer industry was ten years ago. You'll recall that back then the IBM PC had not yet been introduced. Rather, small companies such as IMSAI, Processor Technology and MITS were selling computers primarily in kit form to hobbyists. You had to be a tinkerer to use a microcomputer or to even want to use one.

Apple burst upon the scene in the late 1970s with a "ready to run" computer that just about anyone could use. The rest is history. Computer users were originally hackers (as in "techies," not online burglars), trying to utilize the machines for a specific purpose, while learning about the technology in the process. People interested in home control similarly want to achieve a particular outcome, like turn the sprinkling system off with a phone call when it starts to rain, but they are also interested in the technology of doing.

**Home automation deals with all types of home control systems—security devices, smart appliances, automatic heating, air-conditioning and watering systems.**

Sometimes it seems that a lot of the fun is missing in computing. Sure, discoveries and personal achievements occur with word processors and spreadsheets, but somehow it's not the same as say, automating the entire lighting in your home. This may sound like futuristic pap, but as computers become more prevalent in the home and links are made between them and household devices, it will not only make our lives easier, but it will be fun. That's what computing is all about.

If you're interested, check out a copy of *Electronic House*. Look into the *X-10 System* and MichTron's hardware and software products. Get that ST doing more than cranking numbers, moving blocks of text or playing the hottest new game. I think you will enjoy your ST all the more.

## Seasonal

Over the years I have put together a computer tool kit of odds and ends based on the things I have had to do. To help you get the most out of your ST, I thought I would share these useful aids and some tips in using them. You may already have some of the same tools in your toolbox.

As a true ST user, I just can't be satisfied with the status quo. There always seems to be some new program or gadget that I just have to have. Therefore, I keep a calculator handy for determining how much money I have spent on my ST system. It has a memory function so that I can just recall the previous sum and easily add to it.

I also have a nice Stanley pocket tape measure for precisely measuring the three inches required to raise the ST off the desk in order to then drop it to reseat the chips on the motherboard. I don't use this tool as much as I did during the first year or so I had my ST. I also keep some masking tape handy for taping down those chips. Don't really need it anymore since I had the machine overhauled.

There are a couple of tools that I just know I'll need pretty soon. I have a brand new chip retractor tool for removing the TOS ROM chips when Atari gets around to issuing the new version of TOS. In addition, I also have a chip inserter tool for inserting the new ROM chips. Although these two tools haven't been used yet, I think they'll be needed real soon now.

Here's something that you wouldn't normally expect to find in a tool pouch: an alarm/chronograph/watch. I use it for measuring how long it takes for GDOS to load and signal me when I can start to use the computer after it's finished. Haven't used that baby much since I got *G+Plus*.

There are a couple of things I use when I write programs in ST BASIC. A screwdriver is always handy for tightening the code (actually I have several different sizes of screwdrivers that I use depending on how big the program is). Of course, I keep a flashlight around to help me find the bugs in my programs.

Remember, this month you can't believe everything you read. ■

# Multi

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

# Paint

APPLICATION

By Bob Tedesco

*MultiPaint* is a desk accessory paint program for the Atari ST. With it, detailed pictures can be created while working alongside other GEM applications that support desk accessories. It can be used to enhance the output of a CAD program or language such as ST BASIC, for example, or to just have fun and doodle while word processing. It is designed to work in color or monochrome, in any screen resolution and is compatible with *DEGAS* and *Neochrome* picture files. Functions include draw, point, copy, magnify, fill, text, circle, disc, frame, box, airbrush, palette, lines and rays with many options. A full-screen view is also provided, as well as variable line styles and drawing weights. In addition a "grabber" function is available to grab images directly off of a GEM screen, which can then be manipulated using any available *MultiPaint* function.

You'll find three files entitled MPAINT.ACC, COLOR.RSC and MONO.RSC (along with the Pascal source code) on your ST-LOG disk. Copy MPAINT.ACC to your boot disk along with the appropriate .RSC file for the type of monitor that you are using. Put your boot disk in Drive A and reset your ST. You activate *MultiPaint* by moving to its name in the desk menu and left-clicking with the mouse. You will be presented with the full-screen window that is the main working display.

A.W.D illustration

### The main window

The main working area of *MultiPaint* is a "window" on your picture in memory. It contains, depending on the current screen resolution, a group of colors at the top called the color selector. The color selector will contain 16 colors in low resolution, four colors in medium resolution and two colors (black and white) in high resolution.

To the right of the color selector is the pattern box, which is used to select the current fill pattern. Located next to the pattern box are two buttons labeled *F* and *O*, which stand for function dialog and options dialog, respectively. These buttons are the means for selecting the dialog boxes with the mouse. The window also contains vertical and horizontal scroll bars and arrows for displaying different parts of your picture, and a close box used to exit *MultiPaint* back to the host program. The "info" line at the top is available to describe the current function. Finally, the working area of the window is the area in which you will create your picture.

Please note: The mouse is disabled from moving into the menu area so as to prevent other applications from jumping in and interfering with your picture.

### The color selector

The color selector represents the color palette currently in use on the ST screen and is used to select a color that all *MultiPaint* functions will use. To select, left-click on the color of your choice. A new color can be selected at any time while working within the main window.

### The pattern box

The pattern box is used to select a pattern that all functions that support patterns will use. To select a pattern, point the mouse at the pattern box and left-click. A new pattern will be displayed. To scroll forward through a series of patterns, continuously hold down the left button while pointing the mouse.

Keyboard modifiers are: alternate, which you hold down while left-clicking to scroll backward through the available patterns; control, which you hold down while left-clicking to immediately move to the "solid" pattern; and shift, which you hold down while left-clicking to immediately move to the first of a series of crosshatch patterns.

### The close box and desk menu

Click on the close box or hit Escape to exit back to the host program, making *MultiPaint* inactive. Your picture will be preserved in memory until you are ready to again make *MultiPaint* active by selecting it from the desk menu.

The keyboard modifier is alternate, which you hold down while selecting *MultiPaint* from the desk menu to use the current on-screen colors. The previous *MultiPaint* color palette will be permanently overwritten with the new one. Hold down alternate while exiting *MultiPaint*, and the current color palette will replace the one used by the host program. In this way *MultiPaint* can be used as a means to change the colors that the host program will use while it is running. Use with caution.

**Please note: *MultiPaint* uses up approximately 100K of memory. While it can be run on an ST with only 512K of memory, one megabyte will be necessary for *Multi Paint* to work in conjunction with larger ST programs.**


### The scroll bars and arrows

The scroll bars and arrows work just as they do on the GEM desktop. With them you can reposition your picture within the main window.

### The undo key

The undo key is active as a means to undo the last series of changes made to the picture in memory. Undo will restore the last changes made up until any of the following actions take place:

The user exits *MultiPaint* making it inactive.

The user clicks on the scroll bars or

either of the arrows.

The user selects either the function dialog or the options dialog.

The user moves into full-screen view.

If any of these actions are performed, any changes made up until that point become permanent.

Please note: Undo is not available in full-screen view or while using the magnify function. In addition, while using the copy function, undo will only restore changes made since the last click of the mouse button. Use with caution. Read on for a description of the *MultiPaint* dialog boxes and the full-screen view.

### The full-screen view

The full-screen view is available to work on your entire picture in memory at one time. To enter the full-screen view, right-click on the mouse button, then right-click again to get back to the main window. While in full-screen view, the undo key is not active. In addition, you may enter the full-screen view while using the copy, magnify or palette functions, but they will not operate; they will only work in the main window. While the full-screen view is useful for rendering images on large areas of the picture, the main window is recommended for use most of the time, so that the undo key is always available to correct mistakes.

### The grabber

*MultiPaint* has the ability to "grab" the contents of a window or even the entire screen of the host program. If your application is GEM-based and supports desk accessories, and *MultiPaint* is inactive, you can grab the contents of the front window on the screen (the one with the window title bar highlighted) by holding down the control and alternate keys, plus the left-shift key simultaneously. The screen will briefly switch to show your picture, and the contents of the window will be incorporated into your picture at the same window location as on the application screen. If there is no window opened on your application screen, then the entire screen contents below the menu bar will be transferred to your picture.

Both of these operations are irreversible, so use with caution. In addition, if you hold down the right-shift key along with control, alternate and left-shift, the GEM-screen color palette will be grabbed along with the image, overwriting the previous palette used by *MultiPaint*. Although the grabber is designed to interfere as little as possible with the host program, there may be conflict with the operation of some applications. Experiment first before doing any important work.

The grabber can be used, for example, to program graphics in a language that supports desk accessories (such as ST BASIC), and incorporate the output immediately into *MultiPaint* for further manipulation. It can also be used to modify two- or three-dimensional objects output by a CAD program. These tasks can be accomplished immediately, without the need to save, then load, the images to and from disk.

### The *MultiPaint* dialog boxes

The function dialog and options dialog boxes are the means by which most of the available options, functions and operations of *MultiPaint* are controlled. The function dialog can be activated by pressing the F1 keyboard key, the help key or by left-clicking on the *F* button next to the pattern box in the main window. To activate the options dialog box, press F2 or left-click on the *O* button in the main window. The options dialog is also available by selecting an icon in the function dialog.

You must be working in the main window to use the dialogs; they are not available in full-screen view. To make a selection in either dialog, left-click on the appropriate icon.

### The function dialog

At the top of the function dialog is the function box, displaying all the available function icons of *MultiPaint*. Below and to the left is the draw-mode box, and to the right are the icons for wiping, loading and saving pictures, as well as an icon to access the options dialog. A function or mode selection will not take effect until the dialog is exited (by pressing Return or left-clicking on the Atari symbol in the lower right-hand corner). The wipe, load, save and options icon selections will take effect immediately.

### The function box

Following is a description of each available *MultiPaint* function. All of the functions use the current draw-mode setting.

*Draw:* The draw function allows you to draw by left-clicking and moving the mouse around the screen. The current weight selected will be used as the brush size for all points. The current fill pattern will be drawn if the current weight is larger than the smallest available size.

Keyboard modifiers are: alternate, which you hold down while left-clicking to draw only one point on the screen as long as the left mouse button is held down; and shift, which you hold down while left-clicking to "streak" a series of points as the mouse is moved across the screen.

*Line:* The line function allows you to draw lines of different weights and styles according to the selections made in the options dialog. Left-click and hold to start the line, then "stretch" it out to the desired length and location. Release the button to draw the line on-screen. If the current weight is larger than the smallest available size, the line-style setting will have no effect.

The keyboard modifiers are: alternate, control and shift. Alternate allows you to draw "K lines." Hold down alternate while left-clicking to use the endpoint of the last line drawn as the starting point of the new one. If a new starting point was established with the control key, that point will be used instead.

Hold control down while left-clicking to establish a new starting point for the K line and ray functions. When the mouse button is pressed, the mouse cursor will turn into cross hairs for more accurate placement. When the button is released, the cross-hairs position will be the new starting point.

Hold down either shift key and left-click to draw a line from the current starting point to the current mouse position.

*Text:* The text function allows you to type characters from the keyboard a line at a time and to paste them into the picture. The text size is controlled by the weight settings in the options dialog. These sizes range from the smallest (3), which is almost too small to read, up to (26), which is useful for headlines or banners. There are also various fonts.

To start a text line, click the left mouse button, and the mouse cursor will be replaced by a vertical line the same size as the text to be displayed. Text is then entered by typing at the keyboard. Special ST characters can be displayed by holding down the control key while typing. The backspace key can be used to go back and delete unwanted characters. This line of text can be positioned anywhere on the display by moving the mouse. Finally, left-clicking the mouse button will paint the text into the picture.

Keyboard modifiers are: alternate, which you hold down while left-clicking (before typing your line) to select the

thickened text style; control, which you hold down to select the slanted text style; left-shift, which you hold down to select the outlined text style; and right-shift, which you hold down to select the underlined text style.

Please note: All of the above keys can be pressed in any combination simultaneously to achieve a combined text-style effect. For example, holding down the left-shift and control keys will yield outlined text that is slanted.

*Copy:* The copy function is a flexible "block mover" that allows a section of a picture to be outlined and then repositioned elsewhere on the screen, either as a move operation or a copy operation. It can even be used to grab part of the screen for use as a temporary "brush." To use, hold down the left mouse button to drag out the outline of a box around the part of the picture to be copied. Release the button, and the mouse cursor will then disappear, and a copy of the image will follow the movement of the mouse around the screen.

At this point, two alternatives are available. One is to simply paste the image down at the desired screen location by left-clicking once again. The other is to press and hold down the right mouse button while moving the image around the screen. This has the effect of painting with the entire image as a brush. Release the right button to stop painting.

Please note that while copy is invoked, the undo key will only restore changes made to the picture from the last click of the mouse button. Also, copy does not operate in full-screen view.

The keyboard modifier is alternate. Instead of copying the image from one part of the screen to the other as explained above, an image can be moved. Hold down the alternate key while left-clicking to drag a box around the image to be moved. When the button is released the image will follow the mouse around the screen, and a solid box in the background color will replace the image in its old location.

Please note: Move always transfers images using the opaque draw mode, regardless of what draw mode is actually selected in the draw mode box.

*Airbrush:* The airbrush function allows for the "spraying" of pixels similar to the spraying of paint in an aerosol can. The flow intensity and the spread can be controlled through the options dialog. To use the airbrush, left-click and hold, and a stream of pixels will be sprayed on the screen. Each point sprayed will be the size of the current weight and (if larger than the smallest available size ) drawn in the current fill pattern. See below for a description of the options dialog.

*Fill:* With the fill function you can fill any area of the picture with the current fill pattern in the current color. An area is defined as any part of the picture enclosed by a solid line and/or the edges of the screen or window. Simply point the mouse at the area to be filled and left-click.

*Circle and Disc:* Both the circle and the disc functions allow you to draw either round circles or ellipses. Disc draws circles in the current fill pattern, and circle draws unfilled circles using the current weight and line style if larger than the smallest available size. To use either function, left-click and hold the mouse button to drag the outline of an ellipse on the screen, moving the mouse outward. Release the mouse button to draw the figure into the picture.

Holding control down while left-clicking will cause the disc to be drawn with a solid outline. This does not apply to the circle function. Holding shift down while left-clicking will cause a circle that is equal in width and height to be drawn.

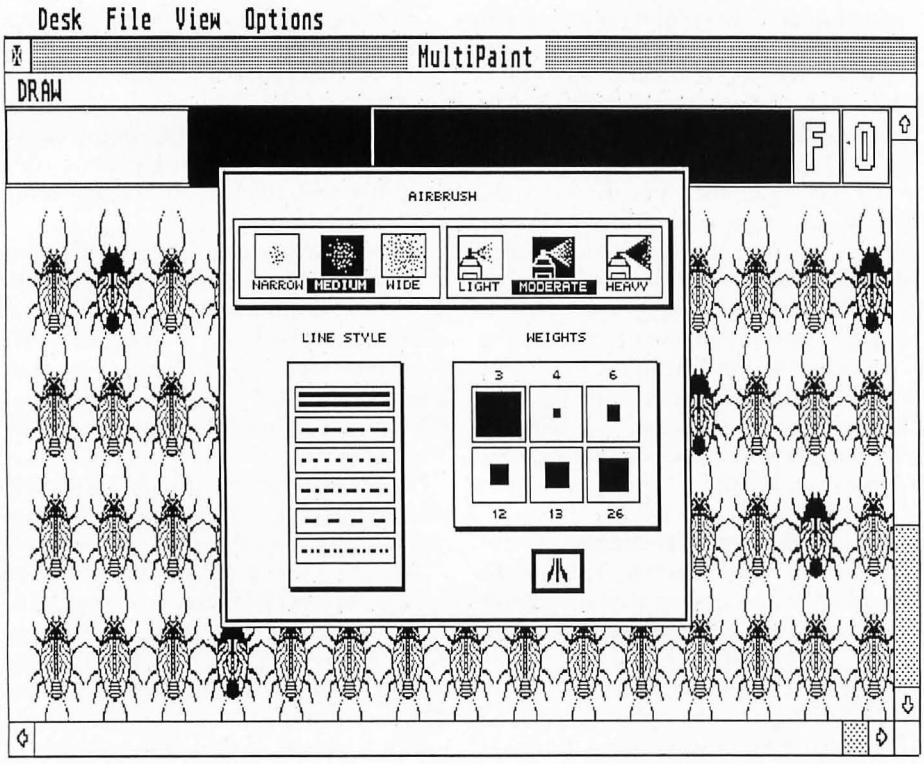
Please note: With the disc function the above keys can be pressed down simultaneously to achieve a filled round circle with a solid outline.

*Frame and Box:* The frame and box functions work just like the circle and disc functions except they do exactly as their names imply. Also, frames and boxes with rounded corners can be created.

Hold down alternate while left-clicking and a frame or box with rounded corners will be drawn. Hold down control while left-clicking and the box drawn will have a solid outline surrounding it. (This does not apply to the frame function.) Hold down either shift key while left-clicking and a square box that is equal in width and height will be drawn.

Please note: The above keys can be pressed down simultaneously in any combination to achieve combined effects.

*Magnify:* Use the magnify function to "zoom" in on an area of your picture for finely detailed work. Each point on the screen will be enlarged to the size of the current weight setting in the options dialog (the smallest weight setting is the same as the next largest size for purposes of the magnify function). Simply left-click to set a point on the screen. The color selector is available as always, to change colors. The undo key is not active while magnify is invoked. While magnify does



not operate in full-screen view, it is still available as a means to see the enlarged image in its original state.

*Palette:* The palette function is the means by which you can change the screen colors to your liking. When palette is invoked, your picture will disappear from the main window and will be replaced by the color panel in the upper left-hand corner (the picture is still in memory; it's just been erased from the window). The color panel has three rows of eight boxes: one each for the red, green and blue (RGB) components or intensities of the currently selected color in the color selector. The first box on the left side of the row represents the darkest intensity for each component; the last box the brightest. The setting for the current color is reflected at all times by the positions of the three RGB markers. Any change in color choice from the color selector will be immediately reflected in the color panel.

To change the currently selected color, left-click on one of the boxes within the RGB component of your choice. The color panel and color selector will both be updated. Alternatively, you can left-click on the RGB letters positioned on the edges of each row, and the RGB marker for that component will be moved in the direction of that letter. The palette function does not operate in full-screen view.

Please note: Palette is not available in high resolution monochrome. To toggle the palette between black and white, press the tab keyboard key.

*The Draw Mode Box:* The draw mode box contains the icons for the X-ray and opaque modes of *MultiPaint*. Each function will use the mode selected in its operation at all times. The X-ray mode causes the background color to be transparent and show through, while the opaque mode causes the background color to block out any image underneath. To illustrate, draw a small frame in the center of the screen; and then on top of it, with the X-ray mode selected, draw a larger box using a crosshatch pattern. The frame should show through the crosshatch pattern of the box. Then select the opaque mode and draw an even larger box with a different pattern on top. The box and frame underneath are overwritten.

In the X-ray mode, images can be made to overlap. Fill patterns can be combined to create new patterns. New color tints and shades can be created if the patterns being overlapped are of different colors.

*The Wipe Icon:* Use the wipe icon to erase your picture from memory. Upon selection, you will be asked to confirm the wipe. This operation is irreversible, so use with caution.

*Load and Save:* The load and save icons are used to load and store images from and to disk. *MultiPaint* normally stores pictures in *DEGAS* format, but you can also load and save in *Neochrome* format if you are working in low resolution color. Upon selection, you will be presented with a file-selector box, where you can type the directory path and filename of the picture you wish to load or save.

Naturally, any existing pictures created with either *DEGAS* or *Neochrome* can be loaded into *MultiPaint* for viewing or modification.

*The Options Icon:* The options icon allows you to immediately access the options dialog. Upon exit from the options dialog, the function dialog will be brought back and made active once again.

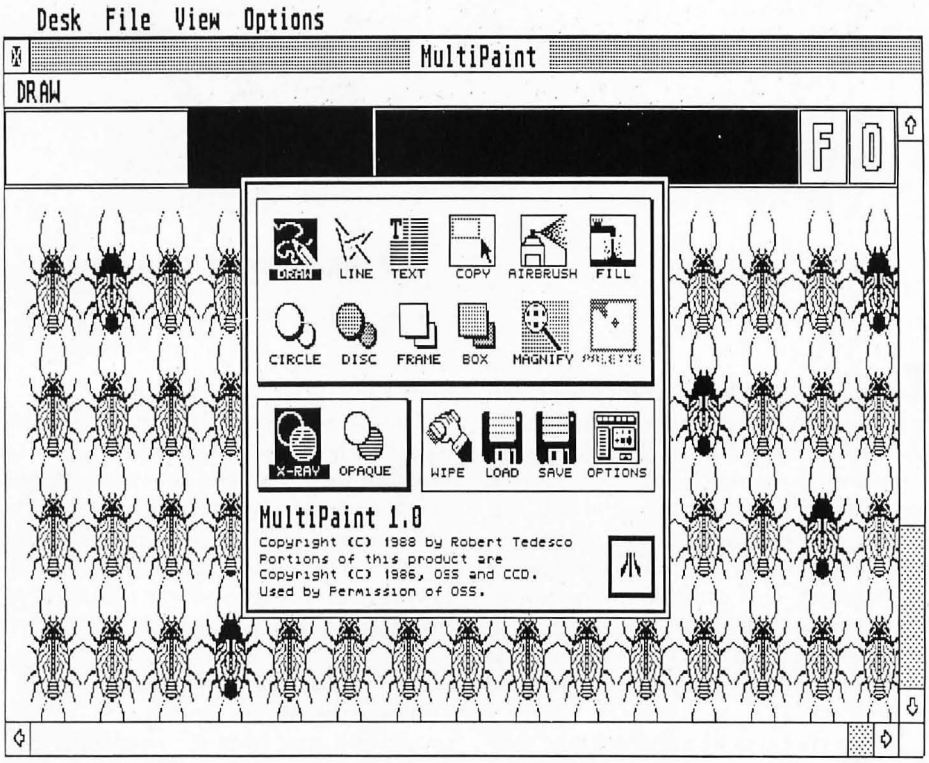
## The options dialog

The options dialog contains control settings for weights and line styles, plus the airbrush. Select your icon by left-clicking the mouse. To exit, click on the Atari symbol in the lower right-hand corner.

*The Airbrush Box:* The airbrush box is composed of three grouped icons on the left, which make up the degree of coverage, and three grouped icons on the right, which make up the degree of saturation.

*The Line Style Box:* The line style box has settings for the various line styles used by the lines, frame and circle functions. If the weight setting is larger than the smallest available size, the line style setting will have no effect (any lines drawn will be solid).

*The Weights Box:* The weights box is the most important control setting in the options dialog. It affects all the functions except copy, fill and palette. Each weight icon illustrates the actual size used by the appropriate function. The numbers shown above and below each icon represent sizes used in connection with the text function. The function dialog descriptions above explain the effect of the weight settings on each individual function.




## That's all he wrote

And that finishes the description of *MultiPaint*'s functions. I hope you will find it a useful addition to your collection.

*Bob Tedesco is a typesetter and freelance programmer from Queens, New York, who has been working with the ST since its introduction. This is his second published program for the ST.*

# ST-CHECK

## A CHECKSUM PROGRAM FOR THE ST

**BY CLAYTON WALNUM**

**ALL RESOLUTIONS**

Typing in a BASIC program listing can be a frustrating and time-consuming task. Just one mistyped character will frequently render a program completely unusable. So to ensure that your program will run correctly, the entire listing must be checked character by character against the original. This can take many hours. To make matters worse, you can't trust your own eyes. Do you know how easy it is to overlook an O where a 0 is supposed to be?

Typing checkers like *ST-Check* take over the arduous task of proofreading your program files. Using this program can cut down your debugging time by a huge factor. When the checker's output matches that published with the listing, you can be sure your typing is accurate.

### Introspection

When you run ST-Check against itself, you will get one of several results. The program may just give up and crash. In that case, go through the listing character by character until you find your typing error.

A second possibility is that the program will run okay, but will create all bad checksum data. This may indicate an error somewhere between Lines 80 and 420. Find the typo and correct it.

The last possibility is that the checksum data will have only a few bad values. In this case, use the normal method detailed below to locate your errors.

*Warning*: Until you get your checksum data for ST-Check to match the data following the listing, you can't trust it to proofread other programs.

### Using ST-Check

When you finish typing a ST BASIC program listing from the magazine, save a copy to your disk, and then run ST-Check. The program will first ask for a filename. Type in the name for the program you wish checked (the one you just saved to the disk), and press RETURN. You'll then be asked for a "bug" name. Enter a filename for the checksum file (this can be any name not already on the disk), followed by RETURN.

ST-Check will now proofread the program. When the checking process is complete, you'll have a file on your disk (saved under your bug name) which contains the checksum data for the program checked.

Check the last value of each line. If it matches the value in the published checksum data, go on to the next. If it doesn't match, you've got a typo.

To find the error, look at the line number of the data statement in which the bad value occurred. This number is equivalent to the first program line the data evaluates. Let's call this "Line X." Count the entries in the data line until you get to the bad value. We'll call this count "Y." Now look at the program you typed in. Starting with and including Line X, count down Y lines. The line you end up on will be the one containing the typo.

Correct the error, and then rerun ST-Check. When you get all the checksum data to match that published in the magazine, your new program is ready to run.

### Passing the buck

Okay, friends. Here's where the truth comes to the fore. I can take only minimal credit for ST-Check, as it's virtually a direct translation from D:CHECK2 (ANALOG #16) by Istvan Mohos and Tom Hudson. All accolades and tribute should be directed to those two fine gentlemen. I'm sure they'll divvy it up fairly, and perhaps pass a small share onto me. Thanks, guys!

You may now type in this month's ST BASIC program, secure in the knowledge that the searching eye of ST-Check is primed and ready.

## ST-Check Listing 1 — ST Basic

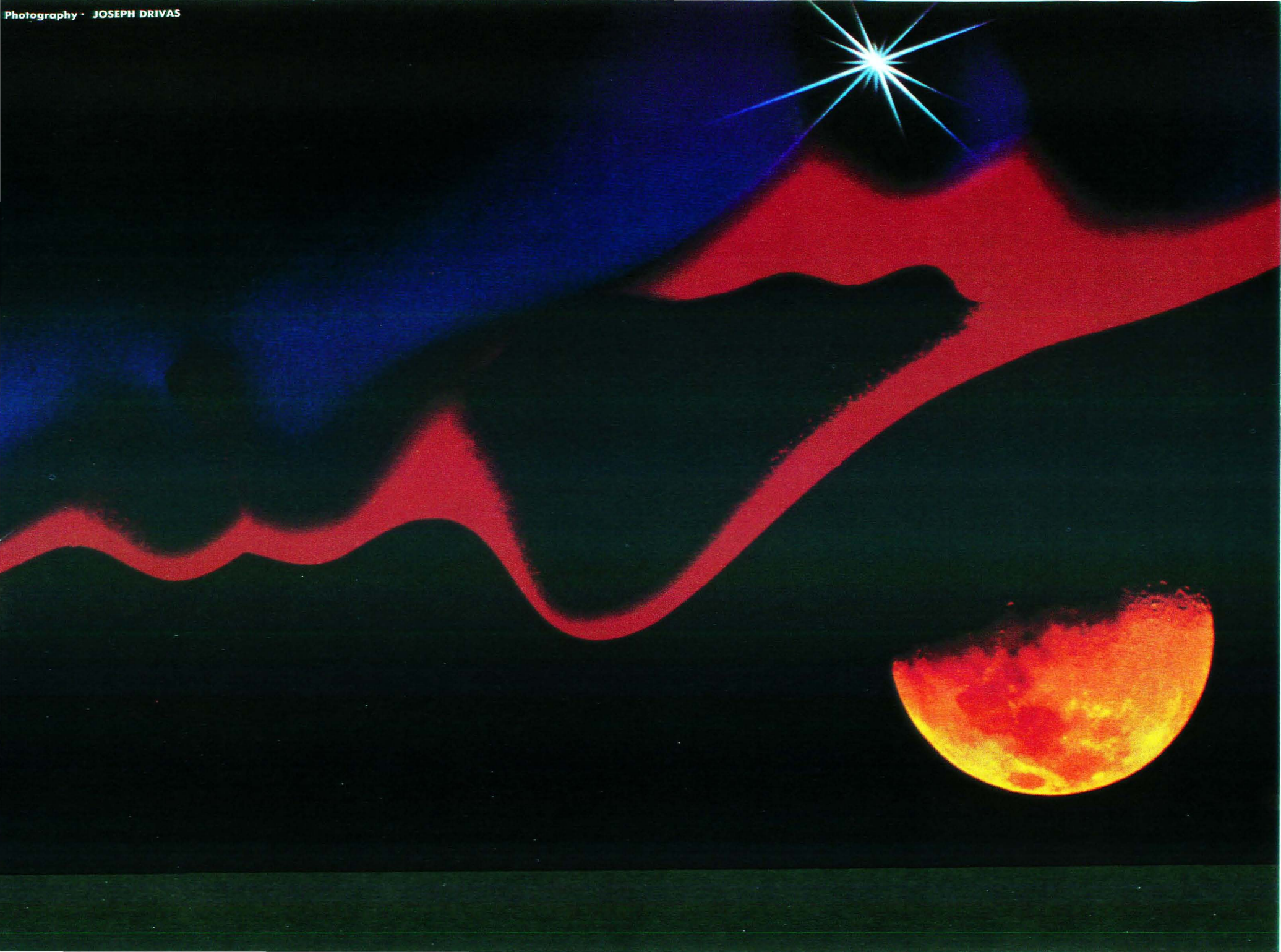
```
10 'ST CHECK typing validator by Clayt
on Walnum
20 'based on a program by Istvan Mohos
 and Tom Hudson
30 if peek(systab)=1 then cl=17 else c
l=32
40 fullw 2:clearw 2:gotoxy cl,0:? "ST
CHECK":ex=0:sp=0:x=0
50 input "Enter filename: ",f$:input "
Enter BUG name: ",f1$
60 on error goto 590:open "O",#1,f1$:o
pen "I",#2,f$:close #2
70 open "I",#2,f$:on x goto 140,220
80 color 2:?:? "Counting lines":lineco
unt=0:color 1
90 on error goto 570
100 line input#2,i$:linecount=linecoun
t+1
110 ? ".";:goto 100
120 close #2:q=int(linecount/10):dim c
(linecount),r(q)
130 x=1:goto 70
140 range=0:lyne=0:color 2:?:?:? "Fill
ing array":color 1
150 ? ".";:count=0
160 line input#2,i$:count=count+1
170 lyne=val(i$):r(range)=lyne:range=r
ange+1
180 on error goto 580
190 line input#2,i$:count=count+1:if c
ount=10 then 150
200 goto 190
210 close #2:x=2:goto 70
220 color 2:?:?:? "Calculating checksu
ms":color 1
240 for i=1 to linecount:checksum=0:li
ne input #2,i$:l=len(i$)
245 if mid$(i$,l,1)=" " then l=l-1:got
o 245
250 for z=1 to l:number=asc(mid$(i$,z,
1))
260 if number=asc(" ") and ex=0 and sp
=1 then goto 320
270 if number<>asc(" ") then sp=0 else
 sp=1
280 if number<>34 then 300
290 if ex=1 then ex=0 else ex=1
300 if ex=0 and number>=asc("a") and n
umber<=asc("z") then number=number-32
310 product=x*number:checksum=checksum
+product:x=x+1:if x=4 then x=1
320 next z:? ".";
330 checksum=checksum-1000*int(checksu
m/1000):c(i)=checksum:x=2:next i
340 close #2:lyne=r(0):item=0
350 color 2:?:?:? "Creating BUG file":
color 1
360 count=10:total=0:if linecount<10 t
hen count=linecount
370 i$=str$(lyne):i$=i$+" data "
380 for i=1 to count:datum=c(10*item+i
)
390 i$=i$+str$(datum):i$=i$+",":total=
total+datum:next i
400 i$=i$+str$(total):print #1,i$:? ".
";
410 item=item+1:linecount=linecount-10
:if linecount<1 then 430
420 lyne=r(item):goto 360
430 close #1:clearw 2:?:gotoxy 0,1
440 ? "To check BUG data against the c
hecksum data found in the magazine,"
450 ? "return to the GEM desktop and d
ouble click your BUG file.  You may"
460 ? "then SHOW the data on your scre
en or PRINT the data to your printer."
:?
470 ? "The line number of each data st
atement coincides with the first line"
480 ? "of the user program the data st
atement evaluates.  Numbers within"
490 ? "each data statement represent c
onsecutive lines of the user program."
500 ? "The last number is the total.":
?
510 ? "Check the last number of each s
tatement against the version in the"
520 ? "magazine.  Only when there's a
discrepancy need you check each number
"
530 ? "in the data statement.":?
540 ? "Take note of the lines containi
ng typos, then make corrections.  When
"
550 ? "all corrections have been made,
 rerun this program to double check."
560 ? "Press <RETURN>":input i$:close
#1:close #2:end
570 if err=62 then resume 120
580 if err=62 then resume 210
590 if err=53 then ? chr$(7);"FILE NOT
 FOUND!":close:resume 50
600 ? "ERROR #";err;" at LINE ";erl:en
d
```

## ST-Check Checksums

```
 10 data  447, 129, 203, 518, 661, 160
, 942, 482, 640, 556, 4738
 110 data  25, 905, 797, 52, 79, 349,
852, 644, 9, 402, 4114
 210 data  883, 479, 834, 822, 42, 498
, 255, 165, 826, 410, 5214
 310 data  337, 1, 166, 578, 136, 801,
 898, 937, 271, 769, 4894
 410 data  363, 99, 155, 889, 243, 764
, 168, 192, 906, 156, 3935
 510 data  757, 251, 146, 509, 146, 91
6, 539, 541, 733, 845, 5383
```

# ST CHECK

END


Photography · JOSEPH DRIVAS

# SOFTWARE ENGINEERING

## The Case for CASE

BY KARL E. WIEGERS

Our exploration of software engineering has revealed that this young discipline includes three main components: methods, procedures and tools. The methods we've discussed include structured techniques for systems analysis and specification, systems design, programming and testing. We really have not talked much yet about the procedures that can help to produce top-quality software, but they include software quality assurance, measurement techniques (metrics) and project management procedures.

In the past few months, we've talked about some "tools" for building models of our systems, defining process specifications and so on. These useful concepts might be classed more appropriately in the methods category, since they define a methodology for modern software development. The tools component of software engineering really refers more to computer software intended to facilitate our task of writing computer software. There are software tools to assist with coding (such as syntax checking editors), documentation (desktop publishing programs, for example), project management, analysis and design and even the automated writing of code.

"Computer-aided software engineering," or CASE, is the term often applied to the idea of using the computer to help us with the steps of program specification, analysis and design, data design and code generation. CASE is the software engineering analog to the computer-aided design, engineering and manufacturing (CAD, CAE, CAM) methods that have been used for years by various sorts of hardware engineers. By now around 100 CASE products are available commercially. It may seem surprising that it has taken so long for software developers to devise programs to help them do their work. But these are very complex software packages, which probably explains why it has taken a while for them to come on the market.

Today I'd like to describe some of the different kinds of CASE tools available and give you an idea of why this is such an exciting subfield of software engineering. CASE has the potential to greatly increase the productivity of individual software developers while also improving the quality of their work. CASE is an extremely important step on the path from computer programming as an art form to genuine software engineering.

## CASE flavors

These are the major categories into which we can classify CASE tools.

*Toolkit*—a group of integrated tools that help you perform one type of software development activity, such as analysis/design, program optimization, code generation and so on.

*Workbench*—a general purpose development environment, probably based on a powerful workstation that contains multiple, integrated tools to assist with the entire range of activities involved in software development. Sort of "cradle to grave" software engineering assistance.

*Methodology companion*—software intended to support a particular software development methodology. "Methodology" here refers to a sequence of steps followed while developing software; there are a variety of different approaches, each with its cadre of enthusiasts and detractors. Methodology companions can be either at the toolkit level or at the more comprehensive workbench level.

As I said, these are not simple programs. Many of them are designed to run on mainframes, minicomputers or powerful graphics workstations. A workstation is a super-microcomputer, usually dedicated to a single user and equipped with a high-resolution graphics display, a fast microprocessor, several megabytes of RAM and a very large hard disk. The Atari Mega ST could be used as the heart of such a workstation, if properly configured. A few CASE packages are available for use on microcomputers such as the IBM PC-AT class, but even these are not cheap, with prices ranging between $1,000 and $10,000. There are even some CASE tools for the Apple Macintosh, but don't hold your breath waiting to see something for the Atari ST line. (Anyone for the Magic Sac?)

Now let's take a closer look at some of the amazing things these products can do for you.

## Analysis and design toolkits

Remember all those nasty data flow diagrams I showed you in previous articles? If you've tried drawing any of them (and I hope you have), I'm sure you quickly discovered how annoying it can be when you have to make a change. You don't want to redraw the whole diagram, so you get out the old eraser. But after several erasures and corrections, you can hardly read the original page, so you need to redraw it anyway. Things are even worse if the change you make affects several diagrams, such as a parent and child pair, not to mention any changes that need to be made in the data dictionary, structure charts or process narratives.

The temptation is to fall into one of two traps whenever you're faced with changing a data flow diagram. The first trap is the tendency to not make the design change at all, thereby compromising the quality of the ultimate program you're writing. The whole idea behind system modeling is to change the system on paper as many times as necessary to wind up with the right final product.

The second trap is the decision to go ahead and change the program, but not modify the corresponding design diagrams. This means you have an inconsistency between your product (the program) and its documentation (the diagrams), thereby reducing the utility of the documentation. Your documentation should always match the system to which it refers. Erroneous documentation is worse than useless: It can waste your time by letting you chase bugs down blind alleys.

Erroneous documentation is worse than useless: It can waste your time by letting you chase bugs down blind alleys.

How can you avoid these traps and still not spend your entire life redrawing circles and arrows and boxes? With a CASE analysis/design toolkit, that's how.

A primary feature of all analysis/design CASE tools is the ability to draw and edit the diagrams used for structured analysis and design. These might include data flow diagrams, structure charts, flow charts, control flow diagrams, state-transition diagrams, entity-relationship diagrams and others (yes, I know we haven't discussed all of these yet). The programs are almost always mouse-driven, using a graphics monitor and on-screen menus to let you select the symbols you want to draw.

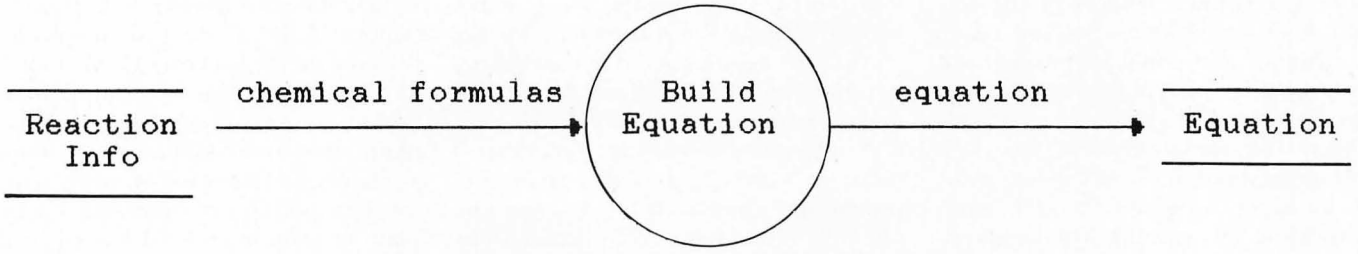
These CASE tools are not unlike paint programs available for the Atari ST, except that they usually have a fixed set of symbols, rules for how you can combine the symbols (no overlapping, for example) and none of the painting kinds of effects, such as free-form drawing, area fill, color and so on. They also have specialized abilities to check diagrams for errors, log entries in a data dictionary and so on. I tried using *Degas Elite* as a poor man's CASE tool for the Atari ST, and it just didn't work out very well. An actual CAD program might work better for the drawing, but it still won't know anything about data dictionaries or rules for DFDs.

Most CASE programs support a variety of diagram-drawing conventions. Some even allow you to define your own methodology and design your own symbols. The DFDs I've shown in previous articles are drawn using conventions usually referred to as Yourdon or DeMarco, named after the software engineering pioneers who conceived these modeling techniques. Other DFD methodologies exist; the shapes and uses of some symbols may vary a little, but the diagrams follow the same general ideas. Many of the software engineering methodologies are named after their founders, so if you want to get your name in the SE history books, here's a possible way to do it.

The real beauty of CASE appears when you need to change a diagram. You can easily add new symbols and connect them to the old, remove symbols and their connecting flows, change symbol sizes and labels and move symbols around on the screen. So, at one level, these CASE tools are graphical editing programs. But there's more; read on.

After drawing a diagram, you may want to validate it to make sure you haven't made any syntactical errors. These CASE programs are smart enough to know the rules for drawing the various sorts of diagrams. They'll keep you honest by warning you of any methodology violations. Examples of such errors on a DFD include failing to name an object, not connecting all objects properly with data flows, having data flows ending up in midair rather than terminating at an object and so on. Unfortunately, the computer can't read your mind, so it can't

FIGURE 1. Sample Data Flow Diagram Fragment.




catch logical boo-boos, only syntax errors.

These CASE programs also know about the relationships between parent and child diagrams. They can "balance" a child against a parent and identify any missing, extra or incorrect parts. In some programs, when you view the child diagram of a process, the flows into and out of that process automatically appear on the diagram for the child.

By letting the computer help handle the mechanical details, you can concentrate on the logical aspects of system design, which is what human beings do best. The computer can remember all the rules we tend to gloss over. It can keep track of the hundreds of detailed components and relationships in a software project with an accuracy we can't approach. It can catch errors we would never see. The result is more effective software development.

## The data dictionary connection

Another feature of the analysis/design CASE programs is the ability to relate the objects in your diagrams to the entries in the system data dictionary. If you've ever tried to thoroughly document a software system, you may have discovered at some later date that your documentation contained inconsistencies, errors or omissions (probably all three). The CASE tool can automatically add new entries to its dictionary, thereby ensuring that you have a complete list at all times. It can insist that you supply a definition for each element before allowing you to proceed with your model building.

The nature of the data dictionary varies from one CASE program to another. Most of them support the conventions I presented in an earlier article called the "DeMarco notation." To refresh your memory, here's an example:

```
reaction = 3{formula + coefficient}4
```

One feature of the analysis/design CASE programs is the ability to relate the objects in your diagrams to the entries in the system data dictionary.

This means that a data flow (or store; we can't tell from this definition) named "reaction" is made up of either three or four occurrences of a combination of an object called "formula," plus an object called "coefficient." See how much more compact the symbolic notation is?

The CASE program can make sure that you don't use a data structure or element on your diagrams in a way that is inconsistent with its definition in the dictionary. The CASE program stores the data dictionary in a database, and you can generate a variety of reports from this information. You might want a list of all the processes which use the data flow labeled "reaction." Or, you might want a list of all the data stores used in the entire system and their definitions. Computers are great for this sort of thing.

## The repository

Remember that one of the goals of software engineering is to reuse as much existing code as we can. Hardware engineers usually build things by combining a bunch of off-the-shelf components in a new way. Why can't we? For one reason, we don't have an inventory of all the existing software components, such that we can see if what we need already exists and just grab a copy if it does. For another, our software components are not produced in standard formats, with standard connections for linking to other software components. In the hardware world, you go buy a MC68000 chip with well-defined proper ties, and you stick it into the right kind of socket with wires coming from the right pins according to published specifications. Not so with software. Some CASE tools are intended to help with these very problems.

All of the information associated with a particular project being developed using a CASE tool is stored in a database or repository. The repository is really the heart of a CASE system. It can contain specifications, documentation, entire diagrams, data dictionary entries and objects that represent the relationships among the components of a diagram.

As an example of this last entity, look at Figure 1. This very simple fragment of a DFD may appear at first glance to contain five objects: data stores called "Reaction Info" and "Equation"; data flows named "Chemical Formulas" and "Equation"; and a process named "Build Equation." But two additional "objects" can be extracted from this diagram. One is the relationship between the three objects Reaction Info, Chemical Formulas and Build Equation; the second is the relationship between the objects Build Equation, Equation (data flow) and Equation (data store).

You already know about reusing actual code; I'm sure you've done it from time to time. But why not reuse a data flow diagram? Or just a part of a DFD? By defining relational objects in the way I just described, we're assembling a catalog of items that could be plucked out of the repository and plugged in wherever they are needed in a future project. This concept of reusability extends well beyond the direct reuse of existing code. By building a system model from a stockpile of existing tried-and-tested components, we are more nearly approaching the enviable situation enjoyed by hardware engineers.

This reusability concept is likely to be one of the real payoffs of CASE, although we really aren't quite there yet. For example, how can we keep track of the objects and relationships already present in our master repository? The computer can define them and store them for us, but how do we find out if the function we need to perform in our new system has already been done before? We could query the database every time we encounter such a relationship, but that's pretty tedious. Wait, I've got it! We'll let the computer query its repository automatically for us. Yeah, that's the ticket!

Some of the most sophisticated CASE tools on the market do include a sort of artificial intelligence system that looks for reusable components in the system being defined. This is a CPU-intensive task, as you might imagine. The situation is akin to having an omniscient system designer peering over your shoulder and pointing out design components he recognizes from earlier projects. Such programs are not yet in widespread use, but someday they will be.

One problem with software developers is that they tend to keep their work to themselves. A consequence is that we constantly reinvent a huge number of wheels, simply because we don't know that someone else has already done the job for us. The multitude of incompatible computer systems doesn't help either. But if a team of programmers all had access to the same master repository of design parts, it would be much easier to reuse existing components. One approach is to have the personal workstations used by individual developers linked to a large computer on which the repository is stored. In the future, I think you'll see more and more professional software development organizations going toward this sort of work environment.

## Code generators

Even with all of the computer assistance that CASE can bring to the analysis and design steps, it still boils down to a human being writing some source code eventually. But does it have to? No.

After the analysis and overview design steps are completed, with the aid of CASE tools to guarantee that your design is valid and internally consistent (note that I didn't say "correct"), you still need to write process specifications before you can type out any source code. A couple of months ago, we looked at several different techniques for writing process narratives, including flow charts, pseudo, code and action diagrams (there are other methods, too). The code you write is only as good as the process narratives you write. If the specs are detailed enough, writing the actual code in the programming language of choice is almost a mechanical task.

Code generators are a kind of CASE tool that automatically produces source code from a set of module specifications.

Suppose your process narratives were written in such a way that a computer program could read them and write the corresponding program code in the correct language? That would certainly save a lot of time, and it would avoid the inevitable typos that creep in because, once again, we're human. As I said, the code still can be only as good as the process narratives. A really smart program could even inspect your process narrative for accuracy and correctness, if it was written in a way that the program could understand. This is the idea behind "code generators." Code generators are a kind of CASE tool that automatically produces source code from a set of module specifications.

CASE tools for analysis and design are sometimes called front-end CASE tools, or "upperCASE." Code generators can be referred to as backend CASE tools or "lowerCASE." The lower/upperCASE nomenclature is probably a feeble attempt at software engineering humor.

There's another kind of code generator idea too. This one consists of a template of code intended for a particular purpose, which is customized according to individual specifications to generate the final source code. An example is a general-purpose program for displaying data-entry screens, providing full-screen editing capability and the ability to validate the user's entries in particular fields according to specified criteria. A template program for this purpose would be merged with specific information about each screen, the fields on it, allowable field values and so on, to produce complete source code.

A number of code generator programs exist at this very minute, ranging from simple Atari BASIC programs that write a subroutine for handling player-missile graphics, on up to mainframe programs that automatically produce valid COBOL source code from appropriate process specifications. The source code produced can either be in the form of a full, executable program, or it might be a skeleton that still needs to be fleshed out manually. For example, the template-type programs I described above may have some provision for you to attach your own individual modules into the generated code. This allows you to take advantage of the computer's program writing ability, while still being able to customize its output to meet your specific needs.

Once a system has entered the maintenance phase, changes undoubtedly will need to be made. If the program was written by a code generator, you have a couple of choices for maintenance. One is simply to forget about the code generator, and just do maintenance on the source code it provided for you (which you may or may not have customized). An alternative is to change your process specifications and regenerate the code for that module, and then re-integrate that module with the rest of the system.

Some CASE products force you into the latter mode, since their code generator directly produces object code, not source code. You couldn't maintain the source code if you wanted to, since there

# 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!*

**$89.95**

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

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 #105 ON READER SERVICE CARD.

isn't any! This is probably a better approach if you have a lot of confidence in your code generator, since you're likely to introduce errors by manually modifying source code while the code generator probably won't. However, this makes it nearly impossible to customize the code generated by the computer.

Code generators have both good and bad aspects. On the plus side, they can remove the tedium and errors associated with manual code creation using a program text editor. They are reproducible: Each time you generate code from the same set of specs, you'll get the same product. The code produced by a particular code generator will have a uniform style, whereas if you gave exactly the same specs to several human programmers, all the programs they write will look different. They can produce some documentation of your system automatically, such as module hierarchy diagrams, lists of variables and their characteristics and data interface information.

On the other hand, the code produced by the generator often isn't very efficient, especially that created by one of the template-type programs I mentioned. Of necessity, such programs are generalized in their purpose, and as such may produce redundant code, introduce variables from the template that you don't need for your specific application and so on. The smarter the code generator, the less of a problem this will be. But a skilled human programmer will nearly always be able to write more efficient code than any automatic code generator.

(Remember, however, that there are several kinds of "efficiency" surrounding computer software: programmer time, computer time and user time. Sacrificing some computer time in order to save programmer time usually is a cost-effective move.)

Code generators that work directly from your specifications may produce more efficient code, but the quality of the code is restricted by the quality of the specification (as always). You won't have the luxury of writing the process narrative any old way you like. You'll have to follow a set of rules intended to let the code generator understand your intentions. You might be able to run the process specification through some kind of automated preprocessor, that will check for syntax and structure problems, before actually producing code.

You've probably already used one variety of code generator: a compiler. A compiler works from specifications (source code) to produce object code. Smart compilers may actually restructure your source code into a more efficient format, although they don't normally tell you what they did so that you could improve the source code yourself. Other CASE tools to optimize and restructure source code for improved efficiency and maintainability do exist for certain languages and computers however.

## Other kinds of CASE

The analysis/design and code generator CASE tools apply generally to the classic software development life cycle we've been exploring in this series. But there are alternative approaches to building software systems (to be covered in a future article), and CASE tools are available to help with them too. For example, another way to build systems is to begin with a partial specification and build a prototype system to address those specifications. By repeated iterations of modifying the specifications and building a new prototype, you can evolve in the direction of the final system.

The good news: CASE is an active subfield of software engineering now. The bad news: CASE programs are not designed to run on Atari hardware systems.

At least one CASE package is available that's designed specifically for this prototyping life cycle. The key to effectiveness in this development mode is rapid turnaround from specification change to creation of the new prototype system. One large chemical company has actually gone into the commercial software business, creating systems for other companies using a CASE environment that permits such rapid interative prototyping. The nature of the projects that can be produced using this technique is somewhat limited, but if your problem falls into that category, it's possible to get a running system in place much faster than using the sequential life cycle we're used to.

Other products touted as CASE tools include so-called screen painters and report generators. A screen painter program helps you design full-screen data entry and menu panels, possibly writing some of the code that drives the panel. A report generator makes it easy for you to design printed reports or screen displays based on such tasks as querying a database. Again, it may actually write the code that will produce the desired formatted output.

Such products sometimes are grouped with other "fourth generation techniques" as another CASE classification. Fourth-generation languages and techniques are software aids that make it easier for a user to program a computer to perform some task, without writing reams of explicit procedural code. Most of the languages you've heard of are procedural (third-generation) languages, like C, FORTRAN, Pascal and BASIC. Fourth-generation languages, or 4GLs, often are tied to relational databases; they are intended in part to make it easier for end users to access the information in the database without needing the services of an official computer programmer.

Still other sorts of CASE software deal with the thorny issues of project planning, control and management. While critical for large-scale software development, they don't apply too much to cottage industry types.

## Getting into CASE

The good news is that CASE is a very active subfield of software engineering at the moment. There are many products on the market, and they are increasing in sophistication as they decrease in price.

The bad news is that most CASE programs are designed to run on a limited number of hardware systems, and Atari isn't one of them. I suspect it never will be. The hardware limitation isn't fatal, but it certainly cramps your style. I've used a PC-based CASE program to design systems that will run on an IBM mainframe computer. This works fine, except that there's no way I could use a PC-based code generator program to produce output that would be useful on a mainframe.

How about the oft-advertised claim of portability of applications written in the C language? Maybe someday we could use a C-code generator (there aren't many yet) on one computer to produce code to be executed on another. Maybe.

If you have access to a computer system for which CASE tools are available, you may be able to use them to design software to run on an Atari. I'm sure most of you don't fit into that enviable category, and most CASE tools are still priced well out of reach for the individual developer, let alone hobbyist. I wish I could be more encouraging.

The bleak future of CASE and the ST doesn't mean you've wasted your time reading this article. Before implementing CASE tools for systems development, you should already have bought into the methodology on which the CASE program is based. Hence, you surely aren't wasting your time by learning these software engineering techniques, such as data flow diagramming, now. You'll be ready to use CASE whenever you have the opportunity this way.

## The future of CASE

Computer-aided software engineering is still in its infancy. There are still large gaps between front-end analysis/design tools and code generators, in most cases. This is equivalent to saying that there are a lot of CASE toolkits available, but not very many CASE workbenches. In the future, more software development will be done in the workstation environment, with each developer having access to an array of powerful CASE tools as well as to a large, shared repository of design components from previous projects. The multitude of computer languages, operating systems and hardware in the world right now will impede progress toward this goal, unfortunately.

Future CASE systems will probably support speech recognition and voice synthesis, which have the potential of being more efficient man/machine communication tools than mice (mouses?) and on-screen messages. CASE programs will become increasingly intelligent, with natural-language processing ability. The goal is to let the computer handle much of the burden of taking abstract design thoughts from us scatterbrained human beings and converting them into the rigorously structured form required for execution by machines.

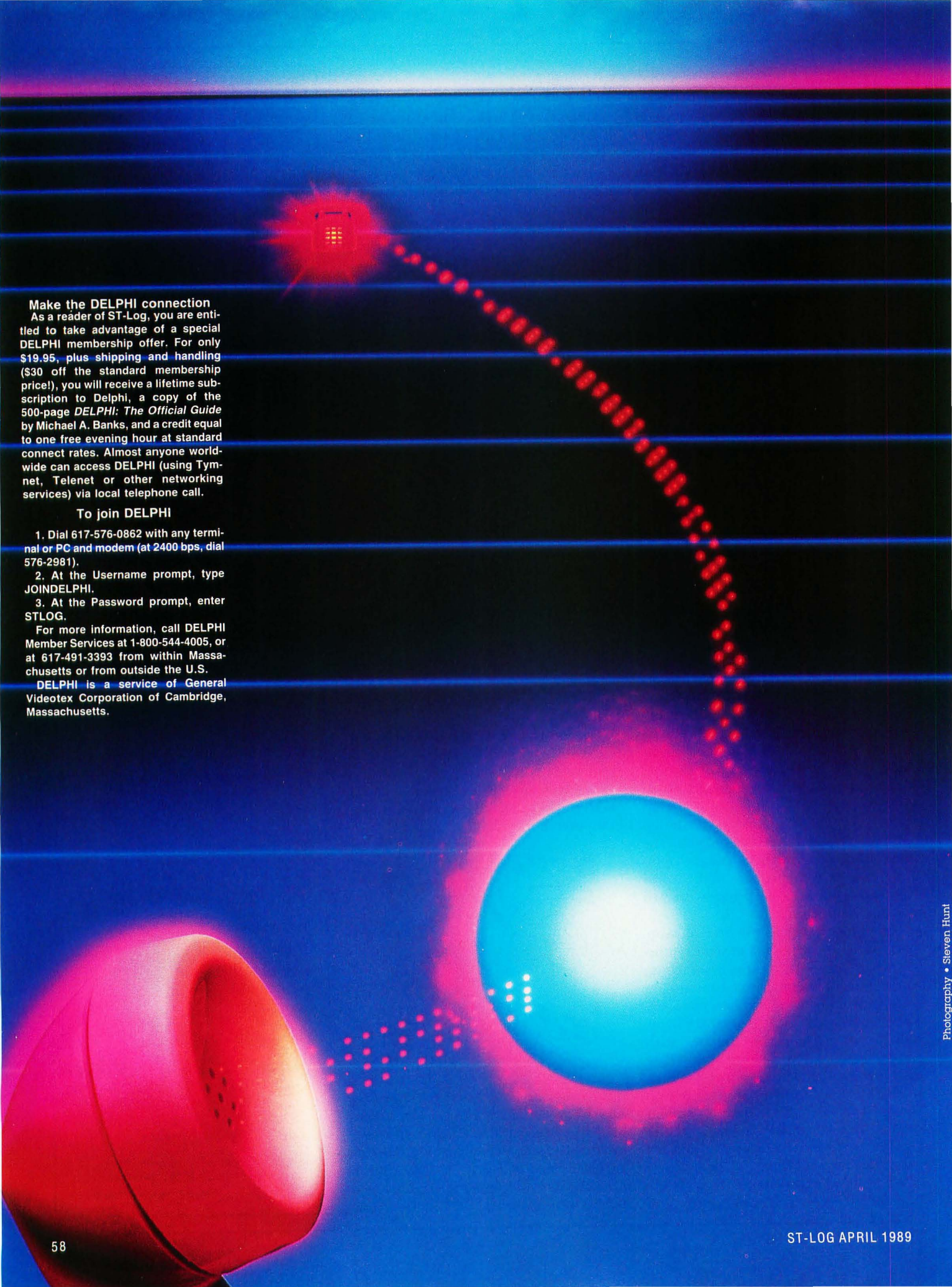
What do we expect to get from CASE? Lots of benefits, including vastly improved quality and much increased productivity. There aren't a lot of data available yet, but the kinds of productivity gains attributable to CASE seem to range between 25% and 2,000%. This is a pretty big range, and it's hard to draw many conclusions from it. The productivity gains depend on how lousy your current development efforts are, just what CASE tools are being used, the nature of the software projects being undertaken, how well-trained the software engineers involved are and even the methods used to measure productivity.

My personal experience suggests that the biggest benefit of using front-end CASE tools is in software quality, not necessarily productivity. The ability to have the computer check your design for errors and inconsistencies is a real step forward, since preventing bugs is highly preferable to searching for them during testing. The real productivity increases will come from increased reuse of existing code and design components.

Automatic code generation offers advantages in both quality and productivity. I'm willing to sacrifice some execution efficiency if someone or something else is writing my program for me, since the net result is a decrease in the time I spend working on the project. As hardware and computer time costs continue to decrease, we can trade computer time for human time.

The short answer is that computer-aided software engineering is probably the best hope for beating the "software crisis" into submission. That makes it pretty important to all but the most casual computer hobbyists.

*After receiving a Ph.D. in organic chemistry, Karl Wiegers decided it was more fun to practice programming without a license. He is now a software engineer in the Eastman Kodak Photographic Research Laboratories. He lives in Rochester, New York, with his wife, Chris, and the two cats required of all* ST LOG *authors.*



## Make the DELPHI connection

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

### To join DELPHI

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

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

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

Photography • Steven Hunt

# Database DELPHI

BY ANDY EDDY

Welcome back to *Database DELPHI*! I hope that April is bringing favorable weather to your locale. As a matter of fact, I'm starting to get a taste of the Southern California climate and what it offers. Now if I could get out of this office, I'd be able to truly enjoy it! Okay, okay, enough complaining from me. Let's get on with this month's discussion.

As I've noted previously, many DELPHI users get on the system to access one or two specific areas (such as ST-LOG's Atari ST SIG). Unknown to them, there is a wealth of other benefits that DELPHI can offer. In fact, there is a section called "Using DELPHI," off the main menu, that holds the key to some of this data. It's the best place to get basic information on the services that DELPHI provides, as well as get information on how much you're using the system.

### Waiter, menu please

The Using DELPHI area can be reached by typing GO USI—all you need to do is type the first couple of letters, so DELPHI knows where you want to go. Figure 1 shows the menu you'll see.

Most of the entries on this menu explain themselves, but we'll go through each selection in order, to give you the full lowdown.

*Advantage Plan*—DELPHI offers a discount program to lower the rates to users who use the system often. Through this program, you can drop the connect charges to as low as $4.80 per hour (as of August 1, 1988). Of course, there are some requirements: You must be a member in good standing, pay a $19, one-time entry fee and make a commitment to use a $24 minimum of connect time each month. This $24 is automatically billed to your account.

In addition to your discount, you'll receive DELPHI's monthly newsletter, a summary of your usage for the month and reduced rates on new versions of the DELPHI handbook and command cards.

*Advice From DELPHI*—As you'd expect, this area contains some basic hints and tips. There are articles on VT52 and VT100 usage, reducing disk storage and changing your username, among others. This is a valuable section that is worthy of perusal.

*Boston Membership Plan*—As we noted in the February 1989 *Database DELPHI* column, there is a service called DELPHI/Boston that is a scaled-down version of the worldwide service with features of interest to Boston users, at the cost of only $9.95 a month—no hourly fees are charged. This offering is profiled here, and membership can be initiated as well.

*Change Address*—This is self-explanatory. To keep DELPHI appraised of any moves you make, you should use this selection. An automatic electronic mail message is sent to SERVICE to notify them.

*Credit Policy*—This notice explains DELPHI'S policy on credit-card usage and invoicing. The most painless way to pay for DELPHI time is by credit card, and they accept MasterCard, VISA and American Express. As well, you can request to be billed by invoice, though a credit application must first be filled out. Each invoice adds a $3.50 charge, but paying in advance will avoid this service charge.

*Feedback*—This selection enables you to send a comment or question to the ones in charge of DELPHI. Another selection on the Using DELPHI menu, *Mail to SERVICE* accomplishes the same thing. Or you can enter the Mail area from most any SIG (type /MAI), and enter SERVICE at the To: prompt.

```
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

USING-DELPHI>(Please Select an Item)>
```

**FIGURE 1**

```
      NEWS ACCU - Accu-Weather Forecasts
    USING ADVAN - Advantage Plan
      ENTER ADV - Adventure Games
      MER CLASS - Advertising, Classifieds
   USING ADVICE - Advice from Delphi
       BUS VES - Advice - Investment
      LIB CAIN - AIDS Information
      TRAV TIPS - Airline information
       TRAV RES - Airline reservations
     NEWS TODAY - Almanac
More?
```

**FIGURE 2**

*Guided Tour*—This is one of the most innovative features of DELPHI, certainly one that should be looked at by other online networks. As the name implies, you'll be led through the basics of the system and how to use it.

*Index*—Many of DELPHI's commands can be abbreviated, and this listing tells you those which are. They are alphabetically listed, as shown in Figure 2.

Many of these areas can be accessed quickly by typing GO in front of it and entering the command from most any prompt. For instance, to get AIDS information, you can type GO LIB CAIN, instead of exiting to the main menu, entering Library and accessing the CAIN (Computerized AIDS Information Network) area.

*Manuals*—This selection is valuable to novice and experienced users alike. Here you can purchase a command card for quick reference, or pick up a copy of the valuable handbook, *DELPHI: The Official Guide*. There are also phone numbers for ordering manuals for Dialog, Lockheed's reference service (documented in the March 1989 *Database DELPHI*) and Digital's EDT Editor, the editor of choice for many DELPHI users for forum and other text duties.

*Member Services*—Simply put, this section gives information on how to contact DELPHI by mail or phone. Figure 3 has that information, which all users should have on-hand, particularly the toll-free number, in case of emergency.

*Network Info*—While you can access DELPHI directly through its Massachusetts line, it's a toll call to most people. For that purpose, packet-switching networks, such as Tymnet and Telenet, play middleman to get you connected, and usually as a local call.

If you want to call directly, this section of the Using DELPHI area advises you how to sign on. Calling directly is much more efficient because there are no delays in data transmission like you get with packet-switching networks.

On the other hand, if you need help accessing DELPHI through an outside network, you can get the number closest to you in this area. First you must pick which network you are interested in: Tymnet, Telenet or Canada's Datapac. Figure 4 shows how you'll be asked where you want the listing to go after you select the network (Tymnet for example).

As you can see, it will tell you the area where the phone number is located, how heavily it is used (sometimes surcharges are added in low-density areas) and if you can use 2400-baud equipment.

### Shut the door on the way out

That's about all the time we have for this month. We'll pick up the Using DELPHI menu again next month.

Before I go, I've received messages from some users regarding storage charges on their DELPHI bill, and now would be a good time to make a quick note about it. As part of your account, DELPHI lets you use some disk space (50 blocks or about 25K) for storage of messages and files. If you exceed that amount, you'll be billed for it—at this writing, they charge 16¢ for every two blocks (1K) above the initial 25K. For this reason, remember to keep your workspace clear of clutter. This doesn't only mean deleting files that don't need to be there, but also keeping your "mailbox" as empty as possible. Long mail messages are saved in your workspace with filenames that start with MAIL$, but if they are deleted in the mail area, they'll also disappear from your workspace. Similarly, you can compress the contents in your mailbox by typing COM (short for COMPRESS), then deleting the MAILOLD file that is created. You should notice your MAIL.MAI file is then within the 50-block ceiling.

Till next month, C U online. . . .

**FIGURE 3**

```
----------------------
Delphi Member Service
----------------------

Voice Hours:

Monday through Friday    -- 8:00 a.m. to 9:00 p.m. EDT
Saturday                 -- 10:00 a.m. to 9:00 p.m. EDT
Sunday                   -- 11:00 a.m. to 12:00 midnight EDT

Phone: 1-617-491-3393 in Massachusetts
       1-800-544-4005 outside Massachusetts

Mail:  Send Delphi Mail to SERVICE.
       Or select MAIL TO SERVICE from the Using Delphi Menu.

Paper Mail: DELPHI; 3 Blackstone Street; Cambridge, MA 02139
```

**FIGURE 4**

```
Search for> los angeles

05241   LOS ANGELES/VERNON    CALIFORNIA      HIGH    213/587-0030
05242   LOS ANGELES/VERNON    CALIFORNIA      HIGH    213/587-0030
06575   LOS ANGELES/VERNON    CALIFORNIA      HIGH    213/587-7514   2400 BPS

Search for>
```

# What exactly *is* an adventure game?

## by Ian Chadwick

YOU ARE IN A LARGE ROOM WITH STRANGE RUNES WRITTEN ALL OVER THE WALLS. THERE IS A JEWELLED AXE ON AN ALTAR. THERE IS A MEAN-LOOKING DWARF HERE. EXITS ARE: NORTH, SOUTH, EAST.
WHAT DO YOU WANT TO DO?
>north

THE DWARF WON'T LET YOU. THE DWARF HITS YOU FOR TWO POINTS.
WHAT DO YOU WANT TO DO?
>south

THE DWARF WON'T LET YOU. THE DWARF HITS YOU FOR ONE POINT.
WHAT DO YOU WANT TO DO?
>east

THE DWARF WON'T LET YOU. THE DWARF HITS YOU FOR ONE POINT.
WHAT DO YOU WANT TO DO?
>kill dwarf

WITH WHAT?
>with axe

YOU DON'T HAVE THE AXE. THE DWARF HITS YOU FOR ONE POINT.
WHAT DO YOU WANT TO DO?
>get axxe

I DON'T SEE ANY "AXXE" HERE. THE DWARF HITS YOU FOR TWO POINTS.
WHAT DO YOU WANT TO DO?
>get axe

YOU NOW HAVE A JEWELLED AXE. THE DWARF HITS YOU FOR TWO POINTS.
WHAT DO YOU WANT TO DO?
>smash dwarf with axe

I DON'T KNOW HOW TO "SMASH" SOMETHING. THE DWARF HITS YOU FOR THREE POINTS.
WHAT DO YOU WANT TO DO?
>kill dwarf with axe

YOU ATTACK THE DWARF. YOU HIT THE DWARF FOR 12 POINTS. THE DWARF HITS YOU FOR 1 POINT. YOU ARE GETTING WEAK.
WHAT DO YOU WANT TO DO?
>get out

I DON'T UNDERSTAND "GET OUT." TRY SAYING IT ANOTHER WAY. THE DWARF HITS YOU FOR 1 POINT.
WHAT DO YOU WANT TO DO?
>exit

DO YOU REALLY WANT TO LEAVE THIS GAME? Y/N
>n

WHAT DO YOU WANT TO DO?
>kill dwarf with axe

YOU ATTACK THE DWARF WITH THE AXE. YOU HIT THE DWARF FOR SIX POINTS. THE DWARF IS DEAD.
WHAT DO YOU WANT TO DO?
>examine dwarf

I DON'T SEE ANYTHING UNUSUAL ABOUT THE DWARF.
WHAT DO YOU WANT TO DO?
>read walls

YOU CAN'T DO THAT—YET!
WHAT DO YOU WANT TO DO?
>Read runes

YOU CAN'T DO THAT—YET!
WHAT DO YOU WANT TO DO?
>push altar

YOU CAN'T DO THAT TO THE ALTAR.
WHAT DO YOU WANT TO DO?
>tilt altar with axe

I DON'T UNDERSTAND "TILT". TRY SAYING IT ANOTHER WAY.
WHAT DO YOU WANT TO DO?
>move altar

THE ALTAR WON'T BUDGE.
WHAT DO YOU WANT TO DO?
>get altar

YOU CAN'T DO THAT TO THE ALTAR.
WHAT DO YOU WANT TO DO?
>inventory

YOU ARE CARRYING THE FOLLOWING:
A BLACK ROCK
A BAG OF SNAKES
A RUSTY DAGGER
A GOLDEN ARROW
AN OLD RUNNING SHOE
A BAG OF MARBLES
LEATHER ARMOR
A JEWELLED AXE
WHAT DO YOU WANT TO DO?
>examine axe

THE AXE IS VERY OLD. THERE ARE MANY JEWELS ON THE POMMEL. THERE APPEARS TO BE WRITING ON THE BLADE.
WHAT DO YOU WANT TO DO?
>read axe

YOU READ THE MAGIC WORDS. THE AXE GLOWS FOR A MOMENT. THERE IS A SUDDEN FLASH AND A MINOTAUR APPEARS.
WHAT DO YOU WANT TO DO?
>west

YOU CAN'T GO THAT WAY. EXITS ARE: NORTH, SOUTH, EAST.
>north

THE MINOTAUR WON'T LET YOU. THE MINOTAUR HITS YOU FOR FOUR POINTS. YOU ARE GETTING WEAK.
WHAT DO YOU WANT TO DO?
>kill minotaur with axe

YOU ATTACK THE MINOTAUR WITH THE AXE. YOU MISS. THE MINOTAUR MISSES.
WHAT DO YOU WANT TO DO?
>south

THE MINOTAUR WON'T LET YOU. THE MINOTAUR HITS YOU FOR TWO POINTS. YOU ARE DEAD. YOU SCORED THREE OUT OF A POSSIBLE 1,024 POINTS. WOULD YOU LIKE TO PLAY AGAIN? Y/N
>argh!

PLEASE ANSWER Y OR N.
>n

GOODBYE AND THANKS FOR PLAYING GARGOYLE'S QUEST IN FUNLAND.

*Argh* is right. Sound familiar? Sort of like the script for a few thousand adventure games you've played? What exactly *is* an adventure game? It's almost easier to say what it's not. It's not quite role-playing (although there is some of that in it, especially in games like *The Pawn* or *Alternate Reality*). It's not fantasy (in the sense of games like *Dungeon Master* and *Questron*, although many adventures have fantasy elements in them). It's not an action game, although some adventures have that too (*King's Quest*).

Adventures are usually text-based (although *Shadowgate* and *Deja Vu* don't require text input). They can have graphics or not (Infocom's successful line of adventures have no graphics at all). They usually follow a particular plot line such as solving a murder (*Deadline*), finding a bride (*King's Quest*) or sometimes simply exploring and surviving (*Zork*, et. al.). Some have no definable plot at all (*Alternate Reality*, although in this case, the story is supposed to evolve with the character and with subsequent releases of the game). The better adventures have complex story lines with multiple characters who can move throughout the environment like you do and respond in a variety of ways. (*Corruption*, an otherwise good game, is sadly flawed by its overt sexism and female stereotypes).

Games like *Star Trek*, *Mercenary* and *Roadwar 2000* are *sort-of* adventures, but I'd rather leave them outside this discussion. Despite adventure elements within them, they're more science fiction/strategy.

Essentially an adventure is a combination of things. It's an interaction between player and game ("the rules of the game are to try and establish what the rules of the game are. . . ."). There are puzzles to be solved (anything from simply figuring out how to open a locked door to solving a murder mystery), a closed environment to explore (often 3-D), items to be found and used (including, sometimes, canards and red herrings). It almost always has a definable end—reaching the exit, solving the crime, running out of time, dying.

Many adventures work on a scoring system that depends on points for objects discovered and/or used, rooms (or other locations) entered and puzzles solved. Better adventures have time-related events, and the game is paced by these events. The secretary sits in her office painting her nails, and you can't get into her drawer until she leaves, which (you learn from frequent playings) is 10:00. So you can say:
>wait until 10:00.

And the game runs itself until that time, or until another event or encounter happens to interrupt the wait.

Encounters with game characters can be as basic and dull as the one parodied before, or very detailed. The characters in *Corruption* respond to different sorts of questions and actions, giving them realistic, 3-D personalities. In many Infocom games, you can converse with characters about other characters. In *Alternate Reality*, how you interact with the characters determines how others will interact with you later and what kinds of encounters you'll have as you progress. You can choose to fight, charm, run, that sort of thing. Unfortunately, the lack of a definable goal in *AR* dulls this otherwise wonderful game.

Puzzles themselves run from the simple to the subtle and complex. They can be a matter of combining the right elements together at the right time (making dynamite in the *Lost Gold Mine* adventure) or a battle of wits (trying to get past the nurse in *Corruption*). You have to immerse yourself in the game and try to think like the characters. Descriptions have clues. Situations have clues. Experimentation is often necessary.

Interaction is through a vocabulary parser. Even the "textless" games like *Deja Vu* have a parser of sorts. They simply

limit the number of choices to a series of general-purpose buttons with labels such as "use," which cover everything from drink, fire, swing, throw, etc.

**Problem No. 1:** small vocabularies. There are some 575,000 words in the English language. The average person has around 5,000 words in his or her working vocabulary. That's not even 1% of what the language has to offer. Professionals, specialists, university graduates, professors and so on average between 15,000 and 20,000 words. Less than 5% of the language, but still three or four times more than most folk.

Most adventures have fewer than 1,500 words. That's like having a conversation with a very young child (or some of the local high-school dropouts). It can be frustrating. My dog is smarter than most adventure parsers. Have you ever spent hours trying to hack out what word the programmer used for a situation? Especially when you *know* there's something there:

>examine painting
I SEE NOTHING UNUSUAL HERE
>look behind the painting
I SEE NOTHING UNUSUAL HERE
>look under the painting
I SEE NOTHING UNUSUAL HERE
>look below the painting
I DON'T UNDERSTAND "BELOW"
>look underneath the painting
I SEE NOTHING UNUSUAL HERE
>look beneath the painting
THERE IS AN ENVELOPE HERE

Originally, adventures were extremely limited. They responded only to a Tarzan/Jane syntax:

>get axe
>kill dwarf
>eat food
>umgowa bwana

Sentences had to be made up of simple verb/object structures, and if there was any need for more information (such as a choice of weapons), the program would come back and ask for more information:

>kill dragon
WITH WHAT?
>use sword

Must have used the *Style Guide for the Terminally Inadequate.* Or maybe that famous textbook, *English as She Is Spoke.*

Infocom broke with this silliness and gave us sentence (not mere word) parsers that could accept complex sentences, sometimes even multiple commands:

>open the drawer, get the key and unlock the door with it. close the drawer. north

Still, adventures have only moved to the idiot savant level, in terms of conversation. They are extremely unforgiving of mistakes and typos. If I type "get the sworrd," *you* know what I mean and recognize the mistake. But if I type it into a game, I get back:

I DON'T SEE ANY "SWORRD" HERE

And I usually lose a turn in the process of making the correction. If I'm under attack by a dwarf or some such denizen, I may be hit and killed before I can enter the right thing. If I forget what exits are available, I lose a turn inquiring or attempting to walk through solid walls.

And what about descriptions? I move into a room and see a long list of information. I can't remember it all. I manipulate things the best I can, but I often have to type "look" to reread the description before I can continue. That costs me turns or time. Sure be nice if I could scroll backward a screen or two through the text to reread what was presented earlier.

Descriptions are key elements. If they're well written, you not only get a good image of what's happening (as well as the ambience of the game), but you find clues and objects. I don't find the graphics in most adventures, except the Mindscape series (*Deja Vu*, etc.) necessary or even interesting. I prefer to depend on the text. I'd trade the disk space a graphic consumes for more complete (and colorful) descriptions any day.

Don't get me wrong. I *like* the idea of adventure games, I like the challenge, the devious twistings of the logic, the puzzles (and I usually hate the vocabulary). I started playing them in 1978 on my TRS-80 and managed to solved most of the original Scott Adams' adventures sometime between '78 and '85. Since then, I've played at quite a few, but never managed to finish any, despite my best efforts to do so.

Why? Read on.

**Problem No. 2:** the time factor. Most adventures take a looonggg time to play. Some are huge: hundreds of rooms to explore, dwarves to pummel, treasures to unearth, puzzles to wrestle into solutions. My own house pales in comparison to some of these places. Many require that you work your way through the game umpteen times, each time gaining a bit more of the map, the location of another item, finding a way out of the maze and so on. I simply don't have the time, even with save-game options. Adventures are "desert island" games: things I'd like to take away with me where I could play through them without distraction from the events of daily life.

There's also the problem of continuity. You have to map things out as you go and usually making notes is wise, so you remember why you crossed the bridge or where you found the axe. You have to play adventures with dedication. You can't leave them sit for weeks, then return to them. You'll have forgotten all the bits and pieces and will probably have to start over again.

In terms of value for the dollar, adventures rate high. You'll probably play them ten times longer than a similarly priced action game. Of course, once solved, an adventure is usually never replayed, unless there are alternate endings you can play for. And adventures provide intellectual challenge far above and beyond the paltry offerings in the arcade game line.

Of the lot, I'd say my favorites (in no special order) are:

*Hitchhiker's Guide to the Galaxy* (Infocom). Bizarre, funny, demanding. Best played after you read the books.

*Corruption* (Rainbird). Fast-paced, tightly written, lots of deviousness and danger. Listen to *both* sides of the tape.

*Deja Vu* (Mindscape). Good interaction, good plot, nice way of handling things.

*Alternate Reality* (Datasoft). Great idea for a game, but unfinished until the series is completed.

Nope, never finished any of them—but I will. One day. I'm working on it. (I have the rest of my life.) And don't send me any clues or hint books. Hint books are like cheating at solitaire. No one wins; you lose. I'll just keep hacking away at them, saving games as I go along. I just wish there were some shorter adventures around. Games I could finish in one or two evenings.

PS: Datasoft and other companies have an annoying habit of writing long, unavoidable introductions to their games (*AR*, *Mercenary*, *Global Commander*). Please, can't you code something to allow players to *stop* the introduction, end the corny music and boring graphics, by pressing a key? I *hate* games where I have to wait several minutes through repetitive silliness before I can even start playing! ■

*Ian Chadwick is a Canadian writer who also plays war games, go, chess and guitar.*

# PD PARADE

by George L. Smyth

This column begins a series of articles that will help you deal with the deluge of public domain (PD) software currently available for the ST series of computers. Each month I will examine one or two of the recent releases, letting you know which programs are the best. Those of you who don't have access to the bulletin boards and online services that are the primary distribution channels for these programs will be pleased to know that the programs being reviewed will be included on the magazine's disk version. Without further ado, let's take a look at a program that will surely become an important part of your software collection.

## Super Boot V5.0

To put it simply, *Super Boot V5.0* is a must for all Atari ST users. Why? One of the problems with the ST is that, at boot time, it doesn't allow the user to configure the current system setup. All accessories on the boot disk are loaded into the desk menu, and all of the programs in the AUTO folder are executed. If the user wishes to load only a specialized group of accessories, a separate disk holding only those accessories must be used to boot the system. DESKTOP.INF files must also be correct for the desired resolution or the system will need reconfiguration. *Super Boot V5.0*, written by Gordon W. Moore, has solved these and many other problems.

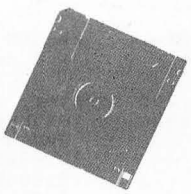
How? The ten function keys are used to determine which setup the user will load. And, with the use of the alternate and control keys, as many as 30 different setup combinations can be designed.

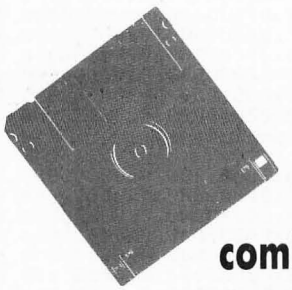
The program used to set up your choices, SUPER__CS.PRG, prompts the user as to which DESKTOP.INF file he/she wishes to use for each configuration. Differing .INF files can be created by modifying the desktop configuration, saving the desktop and renaming the DESKTOP.INF file to a more meaningful name, such as LO__RES.INF, MED__RES.INF or A&D__DRIV.INF. Then, pressing a function key during boot will allow you to load the appropriate desktop setup. Other desktop attributes, such as key click, mouse speed and printer setup, can also be modified, since they are also part of the .INF file information.

All of the commonly used desk accessories can be placed on the boot disk and loaded in preselected combinations by grouping them with SUPER__CS.PRG and assigning the combination to one of the function keys. When you boot your system, you may then choose not only the proper .INF files, but the group of desk accessories you'd like to load as well.

*Super Boot V5.0* accomplishes this by changing the extension of the accessories not selected to .ACX. Although the process takes a bit longer than using separate disks, the convenience far outweighs the additional seconds of load time.

**The ten function keys are used to determine which setup the user will load. And, with the use of the alternate and control keys, as many as 30 different setup combinations can be designed.**





This program has other options—some very helpful, others just fun—that can be executed during system boot. One of these is the ability to alter the date and time. For those without a battery backed-up clock, this handy function is invaluable.

If you have a hard drive, a password option that will help protect your valuable files from access by others is provided. This function will be called only if the system has been cold-booted, so that you will not have to re-enter the password if, for some reason, you had to hit the reset button.

If you are running IBM software on a non-Atari drive, a different seek rate may be required. One of the *Super Boot V5.0* options automatically adjusts this seek rate to match the value needed by your 5¼-inch drive.

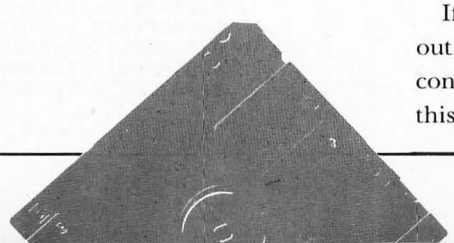
I don't recommend this, but for those who wish to have the speed of their disk write increased, an option has been included in *Super Boot V5.0* to disable the write/verify function. Of course, this means that the system will not make sure that the information written to the disk is correct, but it will more than double the write speed of your drive.

One option I'm particularly fond of is the welcome-screen option. The user is given the ability to take a *Neochrome*, *Tiny* or *DEGAS* picture and have it displayed during system boot. As a matter of fact, you may have one of many pictures for display randomly chosen from a folder.

If for some reason you wish to bypass the *Super Boot* setup routine, this may be done by pressing a "hot key"; a key that is defined during the setup process.

The program comes complete with a 50K instruction manual that takes the user step-by-step through the setup and activation process. Although the program may seem difficult to use the first time, you'll soon become accustomed to *Super Boot*'s setup and implementation.

If your boot disk collection is getting out of hand, or if you want to increase the convenience of your hard drive, check out this month's PD program, *Super Boot V5.0*.



**C-manship**
**Listing 1: C**

```
/*************************************************************
*                    C-manship, Listing 1
*                         ST-Log #30
*                    Developed with Laser C
*************************************************************/
#include <stdio.h>
#include <osbind.h>
#include <gemdefs.h>

#define BLACK       0
#define RED         2
#define WHITE       8
#define DRK_RED     9
#define TRUE        1
#define FALSE       0
#define NONE        0
#define LEFT        1
#define RIGHT       2
#define LOW         0
#define REPLACE     1
#define DOT         2

/* GEM arrays */
int work_in[11], work_out[57], contrl[12], intin[128],
    ptsin[128], intout[128], ptsout[128];

int desk_palette[16]; /* Desktop color palette. */

/* Our own color palette. */
int my_palette[16] = { 0x000,0x700,0x060,0x770,0x007,0x707,0x333,0x666,
                       0x400,0x444,0x373,0x773,0x337,0x003,0x377,0x400 };

int handle,     /* GEM graphics handle */
    dum;        /* A dummy variable for storage of uneeded values */

/* Data for sprites. */
long alien[] = {0x00000000,0x00000000,0x10000000,0x00000800,
                0x00000000,0x00000000,0x20000000,0x00000400,
                0x00000000,0x00000000,0x60000000,0x00000600,
                0x00000000,0x00000000,0x78000000,0x00000600,
                0x00000000,0x00000000,0x00000000,0x00003C00,
                0x00000000,0x00000000,0x30000000,0x00000C00,
                0x00000000,0x00000000,0x30000000,0x00004E00,
                0x00000000,0x00000000,0x42000000,0x00002400,
                0x00020000,0x00000000,0x83400040,0x00404040,
                0x00030000,0x00000000,0x01C00040,0x00408040,
                0x00030000,0x00000000,0x00800000,0x00000040};
int alien_w = 32, alien_h = 10;


long expl1[] = {0x00000000,0x00000000,0x0C000000,0x00000000,
                0x00000000,0x00000000,0x1E000000,0x00000000,
                0x00000000,0x00000000,0x3F000C00,0x00000C00,
                0x00000000,0x00000000,0x3F000000,0x00000000,
                0x00000000,0x00000000,0x1E000C00,0x00000C00,
                0x00000000,0x00000000,0x1E000C00,0x00000C00,
                0x00000000,0x00000000,0x3F000C00,0x00000000,
                0x00000000,0x00000000,0x3F000C00,0x00000C00,
                0x00000000,0x00000000,0x61800000,0x00000000,
                0x00010000,0x00000000,0xC0E00000,0x00000000,
                0x00010000,0x00000000,0xC0E00000,0x00000000};
int expl_w = 32, expl_h= 10;

long expl2[] = {0x00000000,0x00000000,0x18000000,0x00000000,
                0x00000000,0x00000000,0x3C000000,0x00000000,
                0x00000000,0x00000000,0x7E001800,0x00001800,
                0x00000000,0x00000000,0x7E001800,0x00001800,
                0x00000000,0x00000000,0x3C003800,0x00003800,
                0x00000000,0x00000000,0x3C001800,0x00001800,
                0x00000000,0x00000000,0x7E001800,0x00001000,
                0x00000000,0x00000000,0x3C001800,0x00001800,
                0x00000000,0x00000000,0xFF003C00,0x00003C00,
                0x00030000,0x00000000,0x81C00000,0x00000000,
                0x00030000,0x00000000,0x81C00000,0x00000000};

long expl3[] = {0x003C0018,0x00000018,0x00000000,0x00000000,
                0x007E003C,0x0000003C,0x00000000,0x00000000,
                0x007E003C,0x0000003C,0x00000000,0x00000000,
                0x013D0019,0x00000019,0x00000000,0x00000000,
                0x07BD0799,0x00000799,0x00000000,0x00000000,
                0x03FF0199,0x00000181,0x80000000,0x00000000,
                0x01FF01FF,0x000001FF,0xC000C000,0x0000C000,
                0x00FF006D,0x0000006D,0x00000000,0x00000000,
                0x03B90038,0x00000038,0xC0000000,0x00000000,
                0x03AD002C,0x0000002C,0xC0000000,0x00000000,
                0x00260026,0x00000026,0x00000000,0x00000000};
```

```
long expl4[] = {0x01350004,0x00000000,0x00000000,0x00000000,
                0x02CA0282,0x00000082,0x00000000,0x00000000,
                0x01340010,0x00000000,0x80000000,0x00000000,
                0x04B500B1,0x00000031,0x00000000,0x00000000,
                0x0BAA0B00,0x00000A00,0x80008000,0x00000000,
                0x06190210,0x00000210,0x40000000,0x00000000,
                0x00FC00B0,0x00000000,0x80008000,0x00000000,
                0x04A604A0,0x000000A0,0x00000000,0x00000000,
                0x01490009,0x00000000,0x00000000,0x00000000,
                0x07330020,0x00000020,0x80000000,0x00000000,
                0x07030000,0x00000000,0x80000000,0x00000000};

long photon1[] = {0x00000000,0x00000000,0x00C00000,0x00000000,
                  0x00000000,0x00000000,0x00C00000,0x00000000,
                  0x00000000,0x00000000,0x00C00000,0x00000000,
                  0x00000000,0x00000000,0x00C00000,0x00000000,
                  0x00000000,0x00000000,0x00800000,0x00000340,
                  0x00000000,0x00000000,0x00800000,0x00000760,
                  0x00000000,0x00000000,0x01800000,0x00000E70,
                  0x00000000,0x00000000,0x03C00080,0x00000C30,
                  0x00000000,0x00000000,0xFFC00100,0x00000030,
                  0x00000000,0x00000000,0x01F00100,0x00000E00,
                  0x00000000,0x00000000,0x023C0000,0x000005C0,
                  0x00000000,0x00000000,0x041F0000,0x000003C0,
                  0x00000000,0x00000000,0x080F0000,0x00000000,
                  0x00000000,0x00000000,0x00070000,0x00000000,
                  0x00000000,0x00000000,0x00020000,0x00000000,
                  0x00000000,0x00000000,0x00000000,0x00000000};
int photon_h = 15,  photon_w = 32;

long photon2[] = {0x00000000,0x00000000,0x20000000,0x00000000,
                  0x00000000,0x00000000,0x60000000,0x00000000,
                  0x00000000,0x00000000,0x30000000,0x00000000,
                  0x00000000,0x00000000,0x18040000,0x00000000,
                  0x00000000,0x00000000,0x0C080000,0x000003C0,
                  0x00000000,0x00000000,0x04100000,0x000003E0,
                  0x00000000,0x00000000,0x03E00100,0x00000C10,
                  0x00000000,0x00000000,0x03C00080,0x00000C30,
                  0x00000000,0x00000000,0x03C00100,0x00000C30,
                  0x00000000,0x00000000,0x01800000,0x00000E70,
                  0x00000000,0x00000000,0x01000000,0x000006E0,
                  0x00000000,0x00000000,0x01000000,0x000002C0,
                  0x00000000,0x00000000,0x03000000,0x00000000,
                  0x00000000,0x00000000,0x03000000,0x00000000,
                  0x00000000,0x00000000,0x03000000,0x00000000,
                  0x00000000,0x00000000,0x03000000,0x00000000};

long photon3[] = {0x00000000,0x00000000,0x60000000,0x00000000,
                  0x00000000,0x00000000,0x30200000,0x00000000,
                  0x00000000,0x00000000,0x18200000,0x000003C0,
                  0x00000000,0x00000000,0x0C400000,0x000003A0,
                  0x00000000,0x00000000,0x03800000,0x00000C70,
                  0x00000000,0x00000000,0x03C00100,0x00000C30,
                  0x00000000,0x00000000,0x03FF0040,0x00000C00,
                  0x00000000,0x00000000,0x03800080,0x00000C70,
                  0x00000000,0x00000000,0x06000000,0x000001E0,
                  0x00000000,0x00000000,0x1C000000,0x000003C0,
                  0x00000000,0x00000000,0x78000000,0x00000000,
                  0x00000000,0x00000000,0xF0000000,0x00000000,
                  0x00000000,0x00000000,0x60000000,0x00000000,
                  0x00000000,0x00000000,0x00000000,0x00000000,
                  0x00000000,0x00000000,0x00000000,0x00000000,
                  0x00000000,0x00000000,0x00000000,0x00000000};


/*************************************************************
* Main program.
*************************************************************/
main()
{
   appl_init ();
   open_vwork ();
   if ( Getrez () != LOW )
      form_alert ( 1, "[1][This demo must be run|in low resolution][OK]" );
   else {
      do_animate ();
      Setpalette ( desk_palette );
   }
   v_clsvwk ( handle );
   appl_exit ();
}


/*************************************************************
* do_animate ()
*
* Main program loop. Reads the mouse buttons and goes to
* the appropriate functions based on the button presses.
*************************************************************/
```

```
do_animate ()
{
   int repeat, button, x, y;

   set_colors ();
   repeat = TRUE;
   while ( repeat ) {
      button = 0;
      while ( button == NONE )
         vq_mouse ( handle, &button, &x, &y );
      if ( button == LEFT )
         photon ();
      else if ( button == RIGHT )
         repeat = FALSE;
   }
}


/*************************************************************
* photon ()
*
* Performs the photon animation. The function checks for
* a "hit" by getting the color of the screen from the
* current photon position and comparing it to the alien
* ship's color.
*************************************************************/
photon ()
{
   int x, y, pixel, pen, p;
   long *ph[3];

   p = 0;
   ph[0] = photon1;
   ph[1] = photon2;
   ph[2] = photon3;
   graf_mouse ( M_OFF, 0L );
   draw_icon ( alien, 7, 250, 100, alien_w, alien_h );
   x = 20;
   y = 100;
   pen = BLACK;
   while ( pen!=RED && pen!=DRK_RED ) {
      draw_icon ( ph[p], 6, x-6, y-6, photon_w, photon_h );
      evnt_timer ( 10, 0 );
      draw_icon ( ph[p], 6, x-6, y-6, photon_w, photon_h );
      x += 4;
      if ( ( p+=1 ) > 2 )
         p = 0;
      v_get_pixel ( handle, x+19, y+4, &pixel, &pen );
      evnt_timer ( 50, 0 );
   }
   kill_alien ();
   graf_mouse ( M_ON, 0L );
}


/*************************************************************
* kill_alien ()
*
* Performs the exploding ship animation.
*************************************************************/
kill_alien ()
{
   int x, y, rx, ry, i;
   int pxy[2];

   x = 250;
   y = 100;
   vsf_color ( handle, BLACK );

   draw_icon ( expl1, 7, x, y, expl_w, expl_h );
   evnt_timer ( 50, 0 );
   v_circle ( handle, x+20, y+7, 10 );
   draw_icon ( expl2, 7, x+1, y, expl_w, expl_h );
   evnt_timer ( 50, 0 );
   v_circle ( handle, x+20, y+7, 10 );
   draw_icon ( expl3, 7, x+7, y, expl_w, expl_h );
   evnt_timer ( 50, 0 );
   v_circle ( handle, x+20, y+7, 10 );
   draw_icon ( expl4, 7, x+7, y, expl_w, expl_h );
   evnt_timer ( 50, 0 );
   v_circle ( handle, x+20, y+7, 10 );

   x += 13;
   y -= 4;
   vsm_type ( handle, DOT );
   vsm_height ( handle, 1 );
   for ( i=0; i<40; ++i ) {
      rx = rnd ( 16 );
      ry = rnd ( 16 );
      pxy[0] = x + rx;
      pxy[1] = y + ry;
      vsm_color ( handle, WHITE );
      v_pmarker ( handle, 1, pxy );
```

```
        evnt_timer ( 10, 0 );
        vsm_color ( handle, BLACK );
        v_pmarker ( handle, 1, pxy );
    }
}


/*****************************************************************
* draw_icon ()
*
* A general function for drawing icons. It may be used
* in other programs as long as the header file GEMDEFS.H
* has been included at the top of the program. The input
* to the function is the address of the icon's data, the
* desired drawing mode, the X and Y coords at which the
* icon should be drawn, and the width and height of the
* icon. For low resolution only.
*****************************************************************/
draw_icon ( data, mode, dx, dy, width, height )
long data;
int mode, dx, dy, width, height;
{
   MFDB s_m, scr_m;
   int pxy[8];

   s_m.fd_addr = data;
   s_m.fd_w = width;
   s_m.fd_h = height;
   s_m.fd_wdwidth = width / 16;
   s_m.fd_stand = 0;
   s_m.fd_nplanes = 4;
   scr_m.fd_addr = 0;
   pxy[0] = 0;
   pxy[1] = 0;
   pxy[2] = width;
   pxy[3] = height;
   pxy[4] = dx;
   pxy[5] = dy;
   pxy[6] = dx + width;
   pxy[7] = dy + height;
   vro_cpyfm ( handle, mode, pxy, &s_m, &scr_m );
}


/*****************************************************************
* set_colors ()
*
* This function stores the original desktop colors, and
* then installs the program's palette.
*****************************************************************/
set_colors ()
{
   int x;

   graf_mouse ( M_OFF, 0L );
   for ( x=0; x<16; desk_palette [x++] = Setcolor ( x, -1 ) );
   v_clrwk ( handle );
   Setpalette ( my_palette );
   graf_mouse ( M_ON, 0L );
}


/*****************************************************************
* open_vwork ()
*
* This function opens a virtual work station.
*****************************************************************/
open_vwork ()
{
    int i;
    handle = graf_handle ( &dum, &dum, &dum, &dum );
    for ( i=0; i<10; work_in[i++] = 1 );
    work_in[10] = 2;
    v_opnvwk ( work_in, &handle, work_out );
}


/*****************************************************************
* rnd ()
*
* This function is used to get a random number from 0 to
* n-1. Its input is "n" and its output is the random number.
*****************************************************************/
rnd ( n )
int n;
{
   int r;

   r = ( int ) Random ();
   r = abs ( r ) % n;
   return ( r );
}
```

END

**FONT ID EDITOR**
**Listing 1:**
**ST BASIC**

```
100 OPEN"R",#1,"A:ID_EDIT.PRG",16:FIEL
LD#1,16 AS B$
110 A$="":FOR I=1 TO 16:READ V$:IF V$=
="*" THEN 140
120 A=VAL("&H"+V$):PRINT "*";:A$=A$+CH
HR$(A):NEXT
130 LSET B$=A$:R=R+1:PUT 1,R:GOTO 110
140 CLOSE 1:PRINT:PRINT "ALL DONE!"
1000 data 60,1A,00,00,05,68,00,00,00,0
00,00,00,06,84,00,00
1010 data 00,00,00,00,00,00,00,00,00,0
00,00,00,22,2F,00,04
1020 data 4F,F9,00,00,0B,D4,20,3C,00,0
00,0B,EC,90,81,2F,00
1030 data 2F,01,42,67,3F,3C,00,4A,4E,4
41,4F,EF,00,0C,3F,3C
1040 data 00,19,4E,41,54,4F,D0,3C,00,4
41,4D,F9,00,00,06,68
1050 data 1C,C0,1C,FC,00,3A,42,67,2F,0
0E,3F,3C,00,47,4E,41
1060 data 50,4F,41,F9,00,00,06,68,4A,1
18,66,FC,53,48,43,FA
1070 data 03,2E,10,D9,66,FC,4B,FA,03,2
2D,61,00,01,EA,61,00
1080 data 03,0E,4B,FA,04,A2,61,00,01,D
DE,4A,39,00,00,05,D8
1090 data 67,00,01,D0,0C,79,00,01,00,0
00,05,8C,66,00,01,C4
1100 data 41,F9,00,00,05,E8,43,F9,00,0
00,06,68,10,D9,66,FC
1110 data 0C,20,00,5C,66,FA,52,48,43,F
F9,00,00,05,D8,10,D9
1120 data 66,FC,42,67,48,79,00,00,05,E
E8,3F,3C,00,3D,4E,41
1130 data 50,4F,4A,40,6A,24,4B,FA,03,2
29,61,00,01,8A,4B,FA
1140 data 03,67,61,00,01,82,61,00,01,9
9E,4B,FA,04,4C,61,00
1150 data 01,76,61,00,01,72,60,00,01,4
4A,33,C0,00,00,05,68
1160 data 48,79,00,00,06,C4,2F,3C,00,0
00,00,24,3F,39,00,00
1170 data 05,68,3F,3C,00,3F,4E,41,4F,E
EF,00,0C,4A,80,6A,0A
1180 data 61,00,01,7A,4B,FA,02,F2,60,B
B0,61,00,01,70,61,00
1190 data 01,7C,4B,FA,04,01,61,00,01,2
2E,4B,FA,03,51,61,00
1200 data 01,26,13,FC,00,14,00,00,07,0
04,41,F9,00,00,07,06
1210 data 30,3C,00,14,42,18,51,C8,FF,F
FC,A0,0A,48,79,00,00
1220 data 07,04,3F,3C,00,0A,4E,41,5C,4
4F,4B,FA,03,C6,61,00
1230 data 00,F6,A0,09,41,F9,00,00,07,0
04,4A,28,00,01,66,0C
1240 data 4B,FA,03,3E,61,00,00,E0,60,0
00,00,B8,0C,28,00,01
1250 data 00,01,66,2E,0C,28,00,50,00,0
02,67,08,0C,28,00,70
1260 data 00,02,66,1E,42,79,00,00,05,6
66,61,00,01,00,33,FC
1270 data 00,02,00,00,05,66,4B,FA,03,7
75,61,00,00,AA,60,00
1280 data FF,72,70,00,10,28,00,01,41,E
E8,00,02,42,30,00,00
1290 data 61,00,01,54,4A,40,67,A8,B0,7
7C,7F,FF,62,A2,C0,BC
1300 data 00,00,FF,FF,12,00,E0,48,E1,4
49,80,41,33,C0,00,00
1310 data 06,C4,3F,3C,00,02,48,79,00,0
00,05,E8,3F,3C,00,3D
1320 data 4E,41,50,4F,4A,40,6B,00,FE,C
CE,33,C0,00,00,05,68
1330 data 4B,FA,02,D2,61,50,48,79,00,0
00,06,C4,2F,3C,00,00
1340 data 00,02,3F,39,00,00,05,68,3F,3
3C,00,40,4E,41,4F,EF
1350 data 00,0C,4A,80,6A,0A,61,64,4B,F
FA,01,F5,60,00,FE,9C
1360 data 61,5A,4B,FA,02,C1,61,1E,61,3
3C,2E,00,4B,FA,02,E4
1370 data 61,14,4B,FA,02,D2,61,0E,48,4
47,BE,7C,00,15,67,00
1380 data FE,1E,42,67,4E,41,A0,0A,70,0
00,28,4D,10,1C,67,12
1390 data 3F,00,3F,39,00,00,05,66,3F,3
3C,00,03,4E,4D,5C,4F
1400 data 60,EA,A0,09,4E,75,A0,0A,3F,3
3C,00,02,3F,3C,00,02
1410 data 4E,4D,58,4F,2F,00,A0,09,20,1
1F,4E,75,3F,39,00,00
1420 data 05,68,3F,3C,00,3E,4E,41,58,4
4F,4E,75,4B,FA,01,A7
1430 data 61,B4,4B,F9,00,00,05,E8,61,A
AC,4B,FA,02,7C,61,A6
1440 data 4B,FA,01,A1,61,A0,4B,F9,00,0
00,06,C8,61,98,4B,FA
1450 data 02,68,61,92,4B,FA,01,9B,61,8
8C,10,39,00,00,06,C7
1460 data E1,48,10,39,00,00,06,C6,61,6
60,61,00,00,92,61,00
1470 data FF,76,4B,FA,02,44,61,00,FF,6
6E,4B,FA,01,83,61,00
1480 data FF,66,10,39,00,00,06,C5,E1,4
48,10,39,00,00,06,C4
1490 data 61,38,61,6A,61,00,FF,50,4B,F
FA,02,1E,61,00,FF,48
1500 data 61,00,FF,44,4E,75,70,00,0C,1
10,00,39,62,1A,0C,10
1510 data 00,30,65,14,E3,48,22,00,E5,4
48,D0,81,12,18,C2,BC
1520 data 00,00,00,0F,D0,81,60,E0,4E,7
75,41,F9,00,00,06,B8
1530 data 32,3C,00,04,10,FC,00,30,51,C
C9,FF,FA,42,10,32,3C
1540 data 00,04,C0,BC,00,00,FF,FF,48,C
C0,81,FC,00,0A,48,40
1550 data 11,00,06,10,00,30,48,40,51,C
C9,FF,EE,4E,75,4B,F9
1560 data 00,00,00,06,B8,0C,1D,00,30,67,F
FA,53,4D,4E,75,22,3C
1570 data 00,00,05,3E,20,3C,00,00,00,C
C8,4E,42,4E,75,5C,2A
1580 data 2E,46,4E,54,00,1B,45,1B,66,1
1B,70,20,20,20,20,20
1590 data 20,20,20,20,20,20,20,20,20,20,2
20,20,20,20,20,20,20
1600 data 20,46,6F,6E,74,20,49,44,20,4
45,64,69,74,6F,72,20
1610 data 62,79,20,43,68,61,72,6C,65,7
73,20,46,2E,20,4A,6F
1620 data 68,6E,73,6F,6E,20,20,20,20,2
20,20,20,20,20,20,20
1630 data 20,20,20,20,20,20,20,20,20,2
20,20,1B,71,0D,0A,0A
1640 data 00,07,20,43,61,6E,6E,6F,74,2
20,6F,70,65,6E,20,66
1650 data 69,6C,65,21,0D,0A,0A,00,07,2
20,43,61,6E,6E,6F,74
1660 data 20,72,65,61,64,20,66,69,6C,6
65,21,0D,0A,0A,00,07
1670 data 20,43,61,6E,6E,6F,74,20,77,7
72,69,74,65,20,66,69
1680 data 6C,65,21,0D,0A,0A,00,20,48,6
69,74,20,61,6E,79,20
1690 data 6B,65,79,2E,00,20,20,46,6F,6
6E,74,20,66,69,6C,65
1700 data 3A,20,00,20,20,46,6F,6E,74,2
20,6E,61,6D,65,3A,20
1710 data 00,20,50,6F,69,6E,74,20,73,6
```

```
69,7A,65,3A,20,00,20
1720 data 20,46,6F,6E,74,20,49,44,20,2
23,3A,20,00,20,4E,65
1730 data 77,20,49,44,20,6E,75,6D,62,6
65,72,3F,20,28,1B,70
1740 data 52,65,74,75,72,6E,1B,71,3D,6
6B,65,65,70,2C,20,1B
1750 data 70,50,1B,71,3D,50,72,69,6E,7
74,29,20,3E,1B,65,00
1760 data 0D,0A,0A,20,54,68,65,20,66,6
6F,6E,74,20,49,44,20
1770 data 77,61,73,20,6E,6F,74,20,63,6
68,61,6E,67,65,64,2E
1780 data 0D,0A,0A,00,0D,0A,0A,20,57,7
72,69,74,69,6E,67,20
1790 data 6E,65,77,20,66,6F,6E,74,20,4
49,44,20,2E,2E,2E,2E
1800 data 2E,0D,0A,0A,00,20,41,6E,6F,7
74,68,65,72,20,66,6F
1810 data 6E,74,3F,20,28,1B,70,59,1B,7
71,2F,1B,70,4E,1B,71
1820 data 29,20,3E,1B,65,00,1B,48,0A,1
1B,4A,0A,00,1B,6B,1B
1830 data 4A,00,1B,66,00,1B,6A,00,0D,0
0A,00,43,68,6F,6F,73
1840 data 65,20,61,00,66,6F,6E,74,20,6
66,69,6C,65,3A,00,00
1850 data 00,5A,00,00,00,02,00,04,00,0
00,00,00,05,34,00,00
1860 data 05,AA,00,00,05,6A,00,00,05,8
8A,00,00,05,56,00,00
1870 data 05,C8,00,00,06,68,00,00,05,D
D8,00,00,05,1F,00,00
1880 data 05,28,00,02,00,00,00,06,06,2
24,18,28,0C,0A,06,12
1890 data 0C,36,06,0C,38,06,12,18,30,0
0C,3C,0A,14,0C,0C,50
1900 data 2A,16,14,14,08,20,08,40,34,1
10,01,CC,04,04,04,04
1910 data 04,04,04,04,04,00,00,00,00,0
00,00,00,00,00,00,00
1920 data *
```

**FONT ID EDITOR**
**Listing 1:**
**Checksums**

```
100 data 952,544,391,421,536,556,494,8
860,855,808,6417
1050 data 882,913,932,876,732,923,895,
,866,899,769,8687
1150 data 680,694,873,837,822,682,901,
,860,780,805,7934
1250 data 653,705,744,726,60,917,805,9
901,849,784,7144
1350 data 968,991,899,856,775,872,815,
,921,46,48,7191
1450 data 858,749,24,846,952,773,923,8
873,920,18,6936
1550 data 945,846,772,728,573,808,821,
,707,637,824,7661
1650 data 814,755,850,778,847,797,783,
,756,816,858,8054
1750 data 811,791,870,790,864,823,848,
,777,792,800,8166
1850 data 538,668,689,600,717,669,501,
,216,4598
```

**FONT ID EDITOR**
**Listing 2:**
**Assembly**

```
*************************************************
*                                               *
*  Font ID Editor                               *
*                                               *
*  Copyright 1989 Charles F. Johnson            *
*                                               *
*  Written with the Atari MAD MAC assembler     *
*                                               *
*************************************************

* Last revision: Tuesday, December 27, 1988  1:07:28 pm

        .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 number of bytes to shrink down to

        move.l  d0,-(sp)
        move.l  d1,-(sp)
        clr.w   -(sp)
        move.w  #$4A,-(sp)      ; Mshrink
        trap    #1
        lea     12(sp),sp

        move.w  #$19,-(sp)      ; Get current drive
        trap    #1
        addq    #2,sp
        add.b   #65,d0          ; Convert to ASCII
        lea     fntdir,a6
        move.b  d0,(a6)+
        move.b  #':',(a6)+

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

```
        lea     fntdir,a0
.loop:  tst.b   (a0)+           ; Find the end of the pathname string
        bne     .loop
        subq    #1,a0
        lea     fnttxt(pc),a1
.loop1: move.b  (a1)+,(a0)+     ; Tack the search spec to the end
        bne     .loop1

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

fsel:   bsr     aes             ; We're already set up to call fsel_input

        lea     clears(pc),a5   ; Clear everything below the first line
        bsr     print

        tst.b   file            ; Is there a filename here?
        beq     ex2             ; No, exit stage right
        cmp.w   #1,intout+2     ; Click on OK?
        bne     ex2             ; No, let's just forget it

        lea     pathnm,a0       ; Construct a full pathname from the output
        lea     fntdir,a1       ; of fsel_input
.loop:  move.b  (a1)+,(a0)+     ; First, move in the path
        bne     .loop
.loop1: cmp.b   #'\\',-(a0)     ; Search backward for the last backslash
        bne     .loop1
        addq    #1,a0
        lea     file,a1
.loop2: move.b  (a1)+,(a0)+     ; Now tack on the file name
        bne     .loop2

        clr     -(sp)           ; Open for read only
        pea     pathnm
        move    #$3D,-(sp)
        trap    #1
        addq    #8,sp
        tst     d0
        bpl.s   read1           ; No error, skip ahead

noopen: lea     cantopen(pc),a5 ; "Can't open" message
outta:  bsr     print
        lea     hitakey(pc),a5  ; "Hit any key."
        bsr     print
        bsr     getkey          ; Wait for a keypress
        lea     cr(pc),a5
        bsr     print
        bsr     print
        bra     another?

read1:  move    d0,handle       ; Save file handle

        pea     header          ; Read in the first 36 bytes of the font
        move.l  #36,-(sp)       ; header
        move    handle,-(sp)    ; (ID, point size, and font name)
        move    #$3F,-(sp)
        trap    #1
        lea     12(sp),sp
        tst.l   d0
        bpl.s   .read2          ; If no error, skip ahead

        bsr     closf1          ; Close the file
        lea     cantread(pc),a5 ; "Can't read" message
        bra     outta

.read2: bsr     closf1          ; Close it

        bsr     print_info      ; Display the font information

askem:  lea     savpos(pc),a5
        bsr     print

        lea     prompt(pc),a5   ; "New font ID?"
        bsr     print

        move.b  #20,keybuf      ; Maximum 20 characters

        lea     keybuf+2,a0
        move    #20,d0
.loop:  clr.b   (a0)+           ; Clear the input buffer
        dbf     d0,.loop

        dc.w    $A00A           ; Hide the mouse

        pea     keybuf          ; Get a line of keyboard input
        move    #$0A,-(sp)
        trap    #1
        addq    #6,sp

        lea     curoff(pc),a5   ; Turn text cursor off again
        bsr     print
```



# FONT ID EDITOR

```
        dc.w    $A009           ; Show the mouse

        lea     keybuf,a0       ; Pointer to input buffer

        tst.b   1(a0)           ; Any keys input?
        bne.s   cnv             ; Yes, go see what they were

keep:   lea     kept(pc),a5     ; "The font ID was not changed."
        bsr     print
        bra     another?

cnv:    cmp.b   #1,1(a0)        ; One character?
        bne.s   cnv2

        cmp.b   #'P',2(a0)      ; Print?
        beq.s   hard1
        cmp.b   #'p',2(a0)
        bne.s   cnv2

hard1:  clr     device          ; Set printer device
        bsr     print_info
        move    #2,device

        lea     clprom(pc),a5
        bsr     print
        bra     askem

cnv2:   moveq   #0,d0
        move.b  1(a0),d0        ; Get number of characters
        lea     2(a0),a0        ; Pointer to input buffer
        clr.b   (a0,d0)         ; Put a zero at the end
        bsr     decbin          ; Convert the ASCII string to binary

        tst     d0              ; Did we type a valid number?
        beq     keep            ; No, keep the current ID

        cmp     #32767,d0       ; If higher than 32767, keep the current ID
        bhi     keep
        and.l   #$FFFF,d0       ; Mask off upper word
        move.b  d0,d1           ; Save low byte
        lsr     #8,d0           ; Shift upper byte to lower byte
        lsl     #8,d1           ; Shift lower byte to upper byte
        or      d1,d0           ; Mix 'em together
        move    d0,header       ; Store byte-swapped value

        move    #2,-(sp)        ; Open the file again for read/write
        pea     pathnm
        move    #$3D,-(sp)
        trap    #1
        addq    #8,sp
        tst     d0
        bmi     noopen

        move    d0,handle       ; Save file handle

        lea     writenew(pc),a5 ; "Writing new font ID ...."
        bsr.s   print

        pea     header          ; Write the new ID value to the font file
        move.l  #2,-(sp)
        move    handle,-(sp)
        move    #$40,-(sp)
        trap    #1
        lea     12(sp),sp
        tst.l   d0              ; Error?
        bpl.s   closit          ; No, skip ahead

        bsr.s   closfl          ; Close it
        lea     cantwrit(pc),a5 ; "Can't write" message
        bra     outta

closit: bsr.s   closfl

another?:
        lea     more(pc),a5     ; "Another font file?"
        bsr.s   print
        bsr.s   getkey          ; Get a keypress
        move.l  d0,d7           ; Save it

        lea     curoff(pc),a5   ; Turn text cursor off
        bsr.s   print
        lea     clears(pc),a5   ; Clear screen below title line
        bsr.s   print

        swap    d7              ; Get scan code to low word
        cmp     #$15,d7         ; 'Y'?
        beq     fsel            ; Yep, go do it again

ex2:    clr.w   -(a7)           ; Pterm0
        trap    #$1             ; So long, folks
```



# FONT ID EDITOR

# FONT ID EDITOR

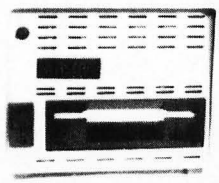
```
* Subroutines

* 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,-(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

closfl: move    handle,-(sp)    ; Close a file
        move    #$3E,-(sp)
        trap    #1
        addq    #4,sp
        rts

print_info:
        lea     fntext(pc),a5   ; "Font file:"
        bsr     print
        lea     pathnm,a5       ; Print the full pathname
        bsr     print
        lea     cr(pc),a5       ; A couple of carriage return/line feeds
        bsr     print

        lea     fname(pc),a5    ; "Font name:"
        bsr     print
        lea     header+4,a5     ; Print the font name
        bsr     print
        lea     cr(pc),a5
        bsr     print
        lea     psize(pc),a5    ; "Point size:"
        bsr     print

        move.b  header+3,d0     ; All values in GEM fonts are stored in 8088
        lsl     #8,d0           ; format (low/high) so we have to switch
        move.b  header+2,d0     ; the bytes around
        bsr.s   bindec          ; Convert the point size to decimal ASCII

        bsr     non_zero        ; Advance the pointer past any leading zeros
        bsr     print           ; Print the point size

        lea     cr(pc),a5       ; Carriage return/line feed
        bsr     print

        lea     idtext(pc),a5   ; "Font ID #:"
        bsr     print

        move.b  header+1,d0     ; Now swap the font ID number
        lsl     #8,d0
        move.b  header,d0
        bsr.s   bindec          ; Convert ID # to decimal ASCII

        bsr.s   non_zero        ; Move past leading zeros
        bsr     print           ; Print the font ID number

        lea     cr(pc),a5
        bsr     print
        bsr     print

        rts

* Convert decimal ASCII to longword binary
*
* Enter with:
* a0 -> ASCII string
```

# Astra gives up !!!

## No we are not going out of business...
## We are going out for business !!!

The demise of many Atari dealers, and the reluctance of many more to carry much inventory forces us to announce:

## Astra Systems goes Factory Direct !



**SYSTEM HD+**

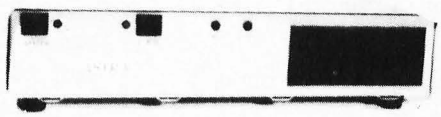
Our original offering for the ST computers is now available in 20, 30, and 40 megabyte capacities. With precision double sided 3 1/2" microfloppy and its tough construction it ranks as the one to beat.

**SYSTEM HD+ 20** $849.95
**SYSTEM HD+ 30** $919.95
**SYSTEM HD+ 40** $1069.95


**RM 60/120 & RM 80/160**

Our rack mount units for the MIDI musician fit portable and permanent racks.

| | |
|---|---|
| RM 60 | $1295.00 |
| RM 60/120 | $2095.00 |
| RM 80 | $1995.00 |
| RM 80/160 | $3495.00 |
| EXPANSION 60 | $ 875.00 |
| EXPANSION 80 | $1595.00 |



**THE EXPANDER**

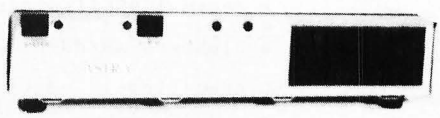
A truly expandable hard disk system. Not a chainable one as many competitors are offering. Contains space, cooling, and sufficient power supply to handle up to 120 megabytes of storage inside the under-monitor case. Built-in surge protection for its 4 AC outlets. Controls the CPU and peripherals from the front of the unit. Optional precision double sided 3 1/2" microfloppy. Twenty, thirty, and forty meg units.

**EXPANDER 20** $759.95
**EXPANDER 30** $819.95
**EXPANDER 40** $999.00

**VISA AND MASTERCARD ACCEPTED**

MASTERCARD VISA

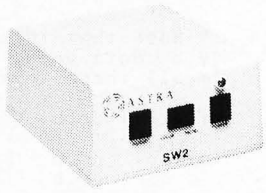
**CALL (714) 549-2141**
**FAX (714) 546-1612**
**BBS (714) 546-5956**



**PRO MODEL**

Four AC outlets control CPU and peripherals from front of unit. Available in 40, 60, and 80 megabyte units. Attractive under-monitor case. Built tough to take it !

**PRO MODEL 40** $899,95
**PRO MODEL 60** $999.95
**PRO MODEL 80** $1499.95




**SW2 MONITOR SWITCH**

Switches both the power and the signal ! True one button switching. Faster and safer. Audio out and composite out (FM models only). Call for information on other exciting MODULAR units.

**SW2** $50.00

**Southern California Astra Systems Inc.**
**2500-L So. Fairview, Santa Ana, CA. 92704-9869**

CIRCLE #107 ON READER SERVICE CARD.

```
*
* Exit with:
* d0 = converted binary number
*
* Clobbers a0/d1

decbin: moveq   #0,d0
.loop:  cmp.b   #'9',(a0)       ; Decimal number?
        bhi.s   notdec          ; No, exit
        cmp.b   #'0',(a0)
        blo.s   notdec
        lsl     #1,d0           ; *2
        move.l  d0,d1           ; Save result
        lsl     #2,d0           ; *8
        add.l   d1,d0           ; *10
        move.b  (a0)+,d1        ; Get character
        and.l   #$0F,d1         ; Make it a number 0-9
        add.l   d1,d0           ; Add to accumulator
        bra     .loop           ; Loop til done
notdec: rts

* Convert binary word to decimal ASCII
*
* Enter with:
* d0.w = binary number to be converted
*
* Exit with:
* decimal ASCII string in 'deciml'
*
* Clobbers a0/d0-d1

bindec: lea     deciml,a0
        move    #4,d1
.zero:  move.b  #'0',(a0)+      ; Initialize string to "0"s
        dbf     d1,.zero
        clr.b   (a0)            ; Set null at end
        move    #4,d1           ; Five digits
        and.l   #$FFFF,d0       ; Just in case
.loop:  ext.l   d0              ; Extend to longword
        divs    #10,d0          ; Divide by 10
        swap    d0              ; Get remainder to low word
        move.b  d0,-(a0)        ; Move it to string
        add.b   #'0',(a0)       ; Make it ASCII
        swap    d0              ; Swap it back
        dbra    d1,.loop        ; Do 'em all
        rts

non_zero:
        lea     deciml,a5
.zero:  cmp.b   #'0',(a5)+      ; Advance a5 to first non "0" character
        beq     .zero
        subq    #1,a5
        rts

aes:    move.l  #aespb,d1       ; Call the AES
        move.l  #$c8,d0
        trap    #2
        rts

* Note that this is still the .text segment.  We don't define
* a .data area, allowing us to use PC relative addressing for
* all these text strings.

fnttxt: dc.b    '\\*.FNT',0

title:  dc.b    '\eE\ef\ep'
        dc.b    '                    '
        dc.b    'Font ID Editor by Charles F. Johnson'
        dc.b    '                    '
        dc.b    '\eq\r\n\n',0

cantopen:
        dc.b    7,' Cannot open file!\r\n\n',0

cantread:
        dc.b    7,' Cannot read file!\r\n\n',0

cantwrit:
        dc.b    7,' Cannot write file!\r\n\n',0

hitakey:
        dc.b    ' Hit any key.',0

fntext:
        dc.b    '  Font file: ',0

fname:
        dc.b    '  Font name: ',0

psize:
        dc.b    ' Point size: ',0
```



FONT ID EDITOR

```
idtext:
        dc.b    '  Font ID #: ',0

prompt:
        dc.b    ' New ID number? (\epReturn\eq=keep, \epP\eq=Print) >\ee',0

kept:
        dc.b    '\r\n\n The font ID was not changed.\r\n\n',0

writenew:
        dc.b    '\r\n\n Writing new font ID .....\r\n\n',0

more:
        dc.b    ' Another font? (\epY\eq/\epN\eq) >\ee',0

clears: dc.b    '\eH\n\eJ\n',0

clprom: dc.b    '\ek\eJ',0

curoff: dc.b    '\ef',0

savpos: dc.b    '\ej',0

cr:     dc.b    '\r\n',0

count1: dc.b    'Choose a',0
count2: dc.b    'font file:',0

        .even

f_sel:  dc.w    90,0,2,4,0

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

addrin: dc.l    fntdir,file,count1,count2

device: dc.w    2

        .bss
        .even

handle: ds.w    1

intin:  ds.w    16
intout: ds.w    16
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
addrout:ds.l    4

file:   ds.b    16

pathnm: ds.b    128

fntdir: ds.b    80

deciml: ds.b    12

header: ds.b    64

keybuf: ds.b    32

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

prgend: ds.w    0

        .end
```

# FONT ID EDITOR

END

# Drafix 1.0

**Foresight Resources Corp.**
**10725 Ambassador Dr.**
**Kansas City, MO 64153**
**(816) 891-1040**
**$195, color or monochrome**

**Reviewed by Ian Chadwick**

***Drafix 1.0*, a 2-D CAD program from Foresight** Resources Corp., comes to the Atari from the PC/MS-DOS world. Like many ports from the IBM environment, features have been trimmed from the original without any apparent additions to counter their loss. And, like so many other programs that make their way to this side of the computer world, ST users have to be satisfied with Version 1.0, since no upgrades are planned. On the IBM side, *Drafix* has gone to "plus" and "ultra" versions, the latest of which has links to spreadsheets, databases, word processors and desktop publishing programs.

The ST version was designed to work as closely as possible to the MS-DOS version. As such, it's somewhat of a hybrid: It only makes full use of some of the ST's features. For example, you cannot access any desktop accessories from within *Drafix*. The program ignores GEM's relatively easy file selector, using its own custom file-selection routines. All function, help and undo keys are ignored (in the IBM version, the function keys work *and* can be altered). Menus don't drop in vertical bands; they lie across the screen horizontally, above the drawing area. While these are simple enough to learn and use, it's irritating, especially when GEM seems perfectly adequate for these functions.

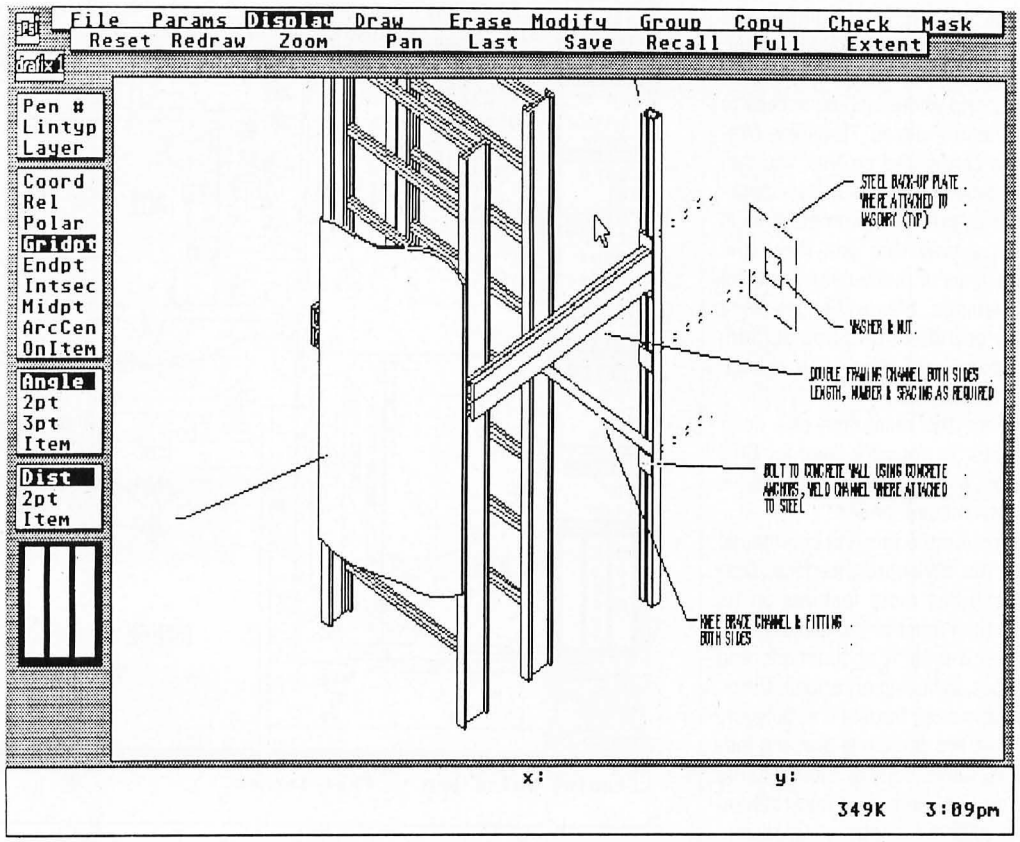
Designed for the professional IBM user, the ST version of *Drafix* sadly drops several of the more powerful commands and features, such as drawing splines and curves, all function key use, including macros, free-hand sketching, stretch individual points, floating text editor, modify font type, font width and text size, default drawing parameters, and copy items with a slant. The ST version has four fewer fonts than the PC version and no resource to design new fonts or information on how to create or port fonts from other applications. One can only assume that Foresight Resources Corp. does not believe ST-based profes-

sionals will need the same features IBM professionals use.

The program comes with three manuals: an ST-specific introduction and installation guide (with considerable information about plotters), Getting Started (an overview and tutorial) and a Technical Reference Guide (all commands explained in menu order and detail). More recently, Foresight Resources has also bundled two separate packages with the main drawing program: *Dot-plot*, required to print your work on a dot-matrix printer, and *General Symbols* (a collection of 400 all-purpose drawing symbols). Three packages of professional symbol libraries are now available, for architectural, engineering and mechanical drafting.

Learning to use *Drafix* requires a thorough reading of the manuals, including the tutorial. Many functions and features work in a convoluted process or perform in unexpected ways. For example, to merge a symbol from the disk to a drawing, you first must select a symbol library (File>Library>Open>Symbol library), then go to the draw menu to select the location of the symbol (Draw>Symbol>Select> From disk) and finally place the symbol on the drawing (Draw>Symbol>Enter). This is hardly an intuitive process. If you look in the index of the Technical Reference Guide, you will not find any indication of where this might be described (that is, you don't already know it's under Symbols, Drawing...). In the Getting Started manual, it's under Symbol, Inserting (a nine-step process, not including the choice of the symbol library).

Although there is nothing inherently difficult in *Drafix*, the unexpected way in which a feature executes may come as a surprise to the user who has not fully read the tutorial. The lack of a simple Undo for features such as modify-stretch and copy>move may well have you pulling your hair out in short order. Without rulers, it's often difficult to position an endpoint or object as precisely as one might wish. The best advice while creating any serious drawing is to save often, after every element, to avoid nasty surprises.




**Learning to use *Drafix* requires a thorough reading of the manuals, including the tutorial. Many functions and features work in a convoluted process or perform in unexpected ways.**

There is no simple "clear screen" command. The "erase all" command removes items, one at a time, apparently in the order they were drawn—a potentially slow process if your drawing is large and complex. I found it easier to save a blank drawing and load it over an existing drawing, rather than wait for a full screen to erase.

No technical information on the file structure is provided, so you can't write your own utilities to link with *Drafix*. This tends to leave *Drafix* by itself at a time when programs like *CAD-3D*/*Cyber-Control* and *DynaCADD* are offering programmers the ability to mesh their output in other applications. The 500 + pages of written material that accompany the program are not always as clear and concise (or informative) as they could be and can be best described as "adequate."

There are no keyboard alternatives or macros to simplify any process, but the most recently selected menu remains "down," so you avoid having to recall menus for common tasks. If, however, you have to switch menus, this structure is tedious and time-consuming. Sometimes the return key works

with input; often it doesn't. The mouse is the only method for commands in most cases.

You have no real control over the color on an SC1224 monitor. There are four choices: blue, red, black and white (and no access to the control panel). However, *Drafix* is configured so that you *can* change colors of the menus, background, grid and prompts, but in such a way that you make the data, lines or menu names vanish! Sometimes, these changes don't have any affect until you perform a subsequent action, such as loading a new file.

Since the resolution of a color monitor is not very good for CAD work, a monochrome display is highly recommended.

Despite the missing commands and the awkward interface, *Drafix* still has more features on its side than most of the competition. It has a wealth of functions and utilities, including grids (on/off/lock, variable size), flexible line, polygon, ellipse, arc and circle drawing features, move, copy or modify items or groups, item or group rotation and alignment, stretch a region, trim lines, bevel, round and fillet a corner, hollow or filled objects, symbols (save, insert, delete, change, explode), text (multiple fonts, at any angle), dimensioning, checking (size, angle, area, distance, symbol name) and masking (selectively viewing objects or symbols).

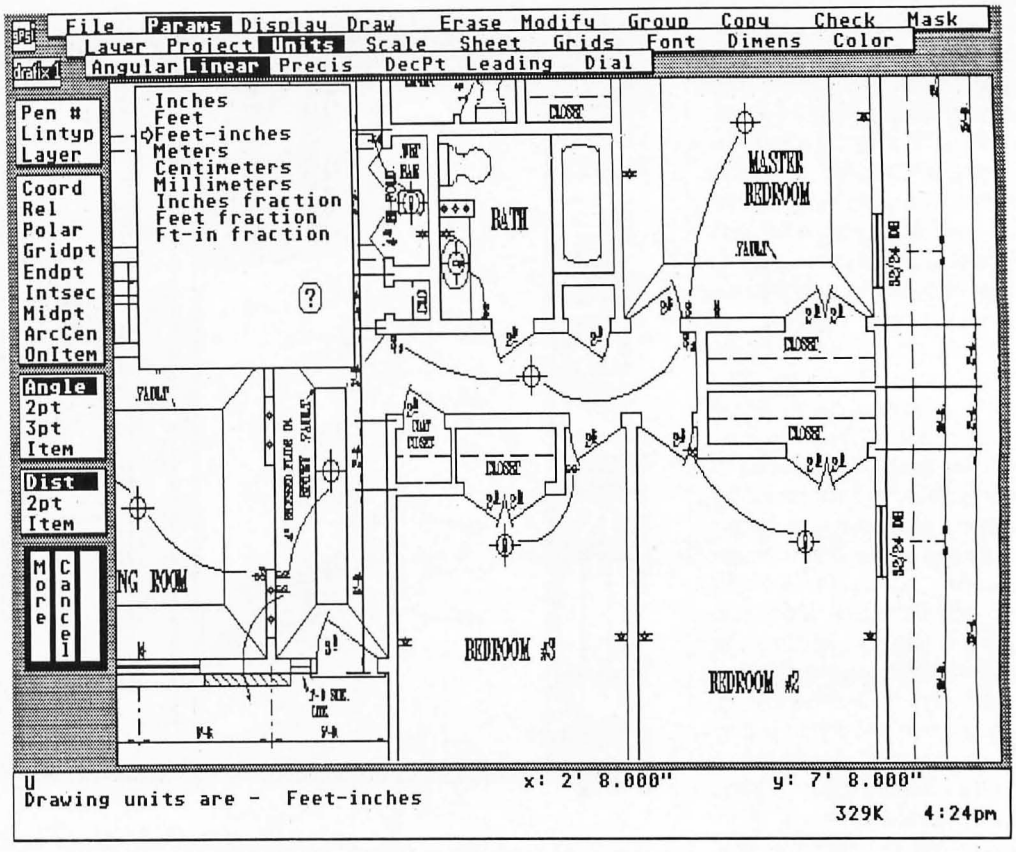
Eight views can be saved for later recall. Zoom uses an easy, dynamic method for specifying the area to zoom into or out of (although there is no means to enter an exact zoom value such as 150%). You can follow a path of zoomed views by clicking on "last," panning a view (again dynamically, without any exact control) or seeing the drawing in a full-screen mode. *Drafix* supports multiple sheet size and a wide variety of plotters and dot-matrix printers. It even saves the workscreen as a *DEGAS*-format file, although annoyingly with a .PIC extension (you'll have to rename it to .PI2 or .PI3 from the desktop).

*Drafix* also fares well in its speed of execution and when redrawing the screen, although large and complex drawings take a while, especially those with text. During the time I worked on this review, *Drafix* proved stable and bug-free. The only times it didn't perform as expected were when I made assumptions as to a function's executions (based on my prior experience with other CAD programs) without reading the manual's explanations first.

Finally, the most recent release of the program comes with a utility to translate files between the IBM and ST worlds.

Overall, *Drafix* could be a great program. It stands up well against *FirstCADD* and *Master CAD*. It has lots of guts inside, but is crippled by what appears to be the company's lackadaisical approach to the ST market. It has an unfinished feel about it—which is about as good as it's going to get without updates or upgrades. Professionals who wish to use the ST as an inexpensive CAD alternative to high-end PC-based systems will find *Drafix* on the ST a disappointment compared to the PC versions. ST users themselves will feel frustrated and annoyed by *Drafix*'s interface and lack of standard GEM features.

In my opinion, ST *Drafix* needs a serious re-think and an overhaul

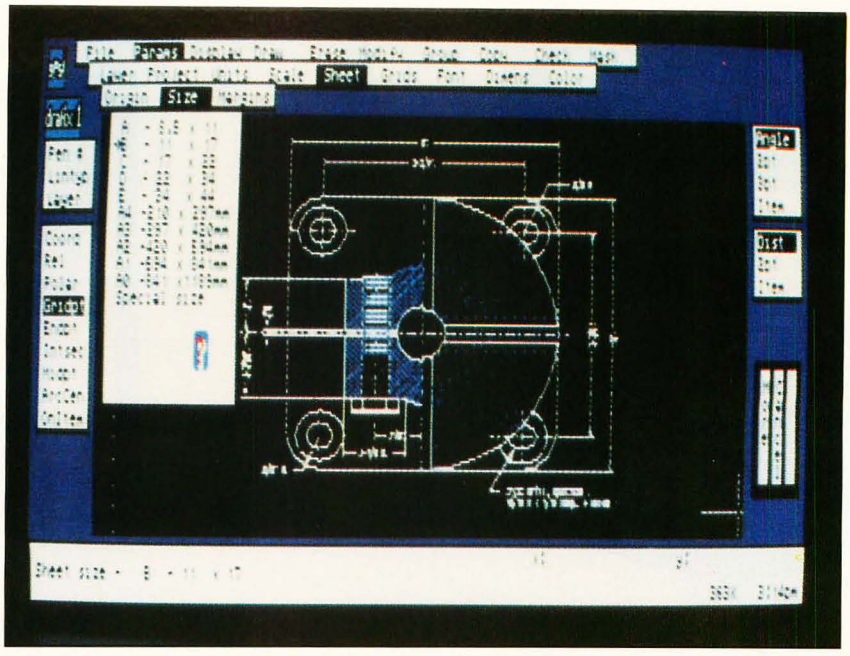



**The best advice while creating any serious drawing is to save often, after every element, to avoid nasty surprises.**

to bring it up to par with the PC version. It should also be made to conform to existing ST standards and make better use of inherent resources, such as GEM. The manual should be improved and enhanced to include proper technical information.

Until Foresight Resources Corp. decides to support the ST market fully and bring out a properly redesigned and upgraded version, *Drafix* runs a distant second to ISD's *DynaCADD* as the professional choice for CAD. ■

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



**Since the resolution of a color monitor is not very good for CAD work, a monochrome display is highly recommended.**

# The ST GAME

## Fire and Forget


**Titus Software**
**20432 Corisco Street**
**Chatsworth, CA 91311**
**(808) 709-3692**
**$39.95, color only**

**Reviewed by Clayton Walnum**

Last month, when I reviewed Titus's *Offshore Warriors*, I mentioned that Titus seems to have a great love of racing games. Now there can be no doubt that this is the case: *Fire and Forget* is yet another game in that genre—the third in a row, in fact.

In *Fire and Forget* you've been chosen by the government to pilot a "Thunder Master"; a vehicle billed as the ultimate weapon. You must take your Thunder Master

# SHELF

into enemy territory and blast away their tanks and gun emplacements, as well as dodge attacking helicopters. The road is filled with all manners of obstacles; some of which you should avoid at all costs and others that you need to acquire in order to keep your fuel tank brimming with Omega Kerosene.

Rather than having a timer like most racing games, *Fire and Forget* uses your fuel tank to control the length of the game. The faster you drive, the sooner you'll find the Omega Kerosene canisters. Every time you're hit by the enemy, your vehicle slows to a crawl—and no matter how fast you're going, you consume the same amount of fuel. Once your tank is empty, the game is over. Also, each time your vehicle explodes, 10,000 points are deducted from your score. If your score goes to zero, the game ends.

The graphics are reasonably nice, as are the sound effects. Unfortunately, game play is hampered by several peculiarities. Sometimes when your vehicle gets hit and explodes, you don't have time to get going before you are hit again. This is further complicated by the fact that if any of the street obstacles are near you when the vehicle reappears on the screen, it will explode once again. Many times, I've sat in front of the computer, gripping the joystick in frustration, while my Thunder Master repeatedly exploded, bringing me prematurely to the end of the game.

Another problem is the animation. As is typical with many arcade-type games on the ST, when the action gets hot, the animation gets jerky. This causes the game to have a sluggish feel, but what is worse, it also means that the graphics can't keep up with the game play. In *Fire and Forget* you are frequently faced with a long line of obstacles that must be blasted out of your way. Unfortunately the fire button can't always keep up with the action on the screen. As a result you find yourself approaching an obstacle at high speed with a gun that doesn't always fire immediately when you press the trigger.

Another problem with the game play is the way the random elements are handled. You know there's a design flaw somewhere when your scores vary from 30,000 to 1,500,000, from one game to the next. The problem seems to be the placement of the fuel cisterns: Sometimes they are placed everywhere, while other times you can't find one no matter how fast you make your vehicle go.

The manual doesn't seem entirely accurate either. It mentions that of the two types of fuel cisterns, only the green ones will refuel your tanks. The blue ones are supposed to destroy your vehicle. Based on my experience, both the blue and green canisters will refuel the tank. And, I don't recall ever exploding when hitting a blue one.

Titus has got a start on what could be a nice game here, but it is only a start and needs a lot of polishing. With some extra programming work, *Fire and Forget* could be a fun game. Without that extra work, you should deal with this game as its title suggests.

**Recommendation: Forget it.**

**Titus has got a start on what could be a nice game here, but it is only a start and needs a lot of polishing.**

# Dive Bomber

**U.S. Gold**
**P.O. Box 8020**
**Redwood City, CA 94063**
**(408) 848-3042**
**$39.95, color only**

**Reviewed by Scott Wasser**

It is May 1941, and the free world is trying to survive the darkest days of World War II. Great Britain is taking an unmerciful pounding from Nazi Germany's war machine. Its cities are being reduced to rubble by Nazi bombs, and its economy is slowly being strangled by Germany's control of the seas surrounding the island nation.

And now the Nazis are about to deliver the final, crushing death blow. Germany launches the mightiest battleship the world has ever seen. With cannons capable of propelling deadly shells as far as 26 miles, the Bismarck has the power to rule the seas and drive Great Britain to its knees.

That's where you come in. Flying a prototype Grumman Avenger dive bomber, you are Great Britain's last hope. *Dive Bomber*, a flight/combat simulation by U.S. Gold puts you in the pilot's seat—and the gunner's and engineer's seats as well. It captures the drama and excitement of military battle as well as a good war movie—maybe even better, since you're actually part of the battle, not just an observer.

The scenario upon which *Dive Bomber* is based is part fact and part fiction. The sinking of the Bismarck did in fact help save the British Empire. But Avengers did not take part in the battleship's destruction. *Dive Bomber*'s instruction manual points that out.

But unless you're a nitpicker for historical accuracy, the fact that there's some fantasy here shouldn't detract from your enjoyment of this program.

*Dive Bomber* is, in fact, one of the best entertainment programs I've ever seen for the ST. The graphics are outstanding; animation flows smoothly and realistically; attention to detail is amazing; and game play is relatively easy to learn, yet intriguing and absorbing enough to ensure hours and hours of entertainment for your software dollar.

The first indication of just how good a job U.S. Gold did comes in *Dive Bomber*'s opening screens. The first one is a picture of an air/sea battle that is so realistic it belongs in some war museum. The image is accompanied by a musical score that gets your heart pounding immediately.

But if the old ticker is a little sluggish, the next two screens should definitely provide the kick-start it needs. The first is a head-and-shoulders portrait of Hitler with a Nazi flag draped behind him. The portrait is accompanied by Hitler's digitized voice proclaiming (in German), "The Bismarck has claimed its first victim. Our navy will crush them!" Then comes a portrait of British Prime Minister Winston Churchill, whose digitized speech proclaims, "We shall fight on until the curse of Hitler is lifted from the brow of men."

It's enough to send shivers down your spine. But try to stay calm, because what you'll have to do to fulfill Churchill's vow will require nerves of steel and a steady hand. It will also require plenty of practice, which you can get by selecting that option from the first *Dive Bomber* menu. That brings up another menu that allows you to practice one of the four flight training elements that comprise an actual mission: takeoffs, landings, combat flight training and attacking the Bismarck.

Whichever training phase is selected, you'll have access to the four control screens used in a *Dive Bomber* mission: the pilot's, engineer's, navigator's and tail gunner's screens. You toggle between the screens by selecting Nos. 1 through 4 from the ST's keyboard.

The pilot's screen shows the instrument panel and the view out the front of the cockpit canopy. There are controls that allow the pilot to steer, climb and descend; fire a .50 caliber machine gun; brake the plane when landing on an aircraft carrier; and load and launch one torpedo. Instruments show heading and attitude of the aircraft, wing flaps and rudder, plus remaining altitude, vertical speed, airspeed and ammunition.

The engineer's screen shows only controls and instruments. This is the screen used to select fuel tanks and monitor fuel supply; control throttle and fuel mixture; monitor engine RPMs, temperature and oil pressure; raise and lower landing gear; and turn on and off the flight camera, ignition switch and landing and cockpit lights.

The navigator's screen is basically a four-quadrant map (one quadrant is visible on the monitor at a time) that shows the location of land masses, mine fields, your aircraft carrier and enemy ships and fighter planes, as well as your dive bomber's location and heading. This screen also features a direction finder. Select your target on this screen and a vertical red bar will appear on the compass in the pilot's screen, showing you which direction you must fly to reach the target.

The fourth and final screen shows the view from the tail gunner's turret. A gauge indicates remaining ammunition, and a gun sight allows you to target incoming enemy fighters. As is the case in the pilot's screen, the gun sight can be moved by joystick or mouse. Pressing the trigger button or left mouse button makes the gun fire. In the pilot's screen, pressing the right mouse button or the spacebar on the keyboard (if you're using a joystick instead of the mouse) toggles control between the machine gun and yoke (which controls the plane's movement).

Contrary to many other flight simulators and combat games, *Dive Bomber* works much better with a mouse than with a joystick. The controls work much more smoothly, and response appears to be more accurate with the mouse. It's also quicker to toggle from machine gun to yoke using the right mouse button than using the spacebar. Generally, I prefer a joystick in this type of game because it just feels like the right tool for the job. But in this case, partly because two of the screens don't really involve maneuvering or shooting and partly because the response to the mouse feels more precise, I didn't mind using the mouse at all.

You might find it difficult at first to undertake a mission. As in flying a real airplane, there are so many controls to monitor and so many elements to control that you need to practice, practice, practice. At the beginning, it helps to play *Dive Bomber* with a friend, who can perform certain tasks for you. It's easier for two people to conquer the split-second tasks. But as those tasks become more routine, it's quite possible for one person to do well at *Dive Bomber*.

Not that it will be easy. Remember, there are four screens to monitor, and neglecting any one of them could put a quick end to your mission. To make things a little easier, each battle station features a "screen selection box," containing numbered squares corresponding to the various battle stations. A square will flash when your attention is required at a station. For example, if you're being attacked from behind, the tail gunner's square will flash, warning you to switch screens.

You'll do plenty of switching during any one of the five possible missions. You "draw straws" to select your mission and have the option of accepting it or drawing for another. Even the simplest mission will be filled with screen-switching, trigger-poking, precision-flying action. All missions are flown at night, so there are only the stars and reflections off the water to indicate your speed and attitude. But the sensation of flying is a good one nevertheless, particularly when you're diving at a target or being attacked by enemy aircraft.

The animation is also a treat. As enemy planes approach, they gradually grow larger, enhancing

**continued on page 89**

# Elite

**Rainbird Software**
**P.O. Box 2227**
**Menlo Park, CA 94026**
**(415) 322-0412**
**$34.95, color only**

## Reviewed by David Plotkin

In *Elite* you take command of a Cobra Mk III spacecraft. She is fast, maneuverable and pretty well armed. Further, because of her versatile construction, weapons and tools can be added to the ship as funds permit. The basic idea of the game is to make a living (and get rich if you can figure out how) as a trader, roaming among 2,000 planets spread across eight galaxies.

The planets are ranked according to the type of government they have (corporate state, dictatorship, anarchy, etc.), the type of economy (medium industrial, poor agricultural, etc.) and the technology level. All of these factors are important. For example, when approaching planets with a stable form of government, you are less likely to be attacked by pirates (did I mention the pirates?). On the other hand, the less stable forms of government offer more chances for a fast buck (ain't it the truth!).

The type of economy comes into play when deciding what to ship and where to ship it to. Agricultural goods will be more reasonably priced in an agricultural world and will fetch a better price in an industrialized world. Further, agricultural worlds have little use for highly advanced technological goods, and so will offer little for them.

The technology level is important because certain items are only available at worlds with a technological rating. A good example is the expensive galactic hyperdrive, which is necessary to proceed to the next galaxy and can only be obtained at worlds with a technology rating of 10 or higher.

Playing *Elite* is relatively simple. You begin the game at a space station orbiting the planet Lave (you don't actually land on planets, just dock with their space stations), with 100 credits to your name. You can then check what's available at the station and use the money to buy items you think you can get a good price for at one of the worlds within seven light-years (your range in space). You can call up a galactic map (not real helpful) or a local map, which shows the nearby planets. Placing the cursor on the planet you want to know about and selecting the "Planet" item from the control panel will tell you important information about that destination.

Then it's time to launch into space, where you can practice docking with the station (not a non-trivial maneuver, these babies

**continued on page 90**

# Rings of Zilfin

**Strategic Simulations, Inc.**
**1046 N. Rengstorff Ave.**
**Mountain View, CA 94043**
**(415) 964-1353**
**$39.95, color only**

## Reviewed by Kevin Peck

*Rings of Zilfin* is an animated fantasy/adventure game based on the save-the-world-from-some-indestructible-overlord-who-plans-to-plunge-the-world-into-darkness theme. The game box includes three single-sided, nonprotected disks, a player's manual, a hint sheet for the beginning player and a sheet to explain the special features of the ST version.

The graphics, while not breathtaking, are detailed and varied, making good use of the 16 colors available in the low resolution mode of the ST. All of the towns and villages have a distinct look and their own set of inhabitants and rumor mills. The roads between the towns are rather monotonous, but there are plenty of hint-bearing monks to speak with, creatures to kill and mushrooms to pick.

There are a number of special animated sequences that make you feel as if you are living in the game world. You must "earn" the right to see these by completing special tasks or properly using the objects that you obtain along the way.

The downside of the graphics appear during combat. There are not enough frames of animation for the characters, causing the animation in this phase to look jerky at best. I also noticed "sparkles" on the screen while walking down the scrolling road—not terrible, but visibly wrong nonetheless.

You can play the entire game with the intuitive mouse interface. A window on the right side of the screen contains a list of all valid commands. Moving the mouse up and down highlights your selection, and a press of the left button executes the command. The right mouse button is used to access a secondary menu, allowing you to view the map, save the game or quit the game.

I ran into one small mouse-related problem during the walking phase of the game. When you move the mouse up and down to select an item on the menu, you can accidentally move it left or right, thus changing your pace. But this is a small problem that is nearly negated by the status reports on your speed given in the action area at the bottom of the screen.

Keyboard equivalent commands for all mouse actions exist. Press the first letter of the command, and it will be executed. The only oddball keystroke is having to press the insert key when the screen states, "Press RETURN to continue." The insert key acts like the left mouse button. It would have made more sense to say,

**continued on page 92**

# Starglider II

**Rainbird**
**P.O. Box 2227**
**Menlo Park, CA 94026**
**(415) 322-0412**
**$44.95, color only**

**Reviewed by Scott Wasser**

Alright, I'm sorry. You can blame me if you look out your window one night and see chunks of the planet Novenia whirling around in the inky sky.

It's my fault. I could have stopped the destruction before it started. But I blew it. I allowed the evil Egron empire to build that star-powered plasma projector that blasted Novenia into more pieces than a porcelain doll in a blender.

Don't ask me how it happened. One minute I was sitting there in front of my ST booting up a copy of Rainbird's *Starglider II*. The next thing I know, it's up to me to save the universe. Or at least the part of it known as the Solice and Novenian star systems.

Save a couple of solar systems? Me? Come on! Who do I look like, John Wayne?

Johnny Carson would be more like it. What do I know about flying a police patrol craft called the Interplanetary Combat and Reconnaissance Universal Scout (ICARUS)?

To make matters worse, Rainbird really never told me exactly what I was supposed to do. Oh, the company definitely provided me with plenty of documents to look at. There was a nifty little novella by James Follett that brought me up to date on the original Starglider saga and explained the new nastiness that the Egron empire was up to.

There also was a Mission Briefing Dossier that provided me with detailed information about the planets and moons comprising the Solice system and told me all about the ICARUS' creation, explained its features and described its control panel.

Finally, there was a key guide that showed me how I could control the ICARUS by using my ST's keyboard and a mouse or joystick.

But being the dimwit that I am, I would have liked a little more than that. I craved something that would spell out for me in no uncertain terms what my mission objective was and exactly what I needed to do to carry it out. Certainly some people would consider such instructions a case of stating the obvious. But I prefer to have the obvious pointed out to me when the fate of the universe is resting on my shoulders.

As it turns out, my shoulders weren't up to the task. I struggled to keep the ICARUS flying and save Novenia and the Solice system from Egron evil. Judging by the

**continued on page 91**

# Tetra Quest

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

**Reviewed by Clayton Walnum**

The six legendary phoenix tablets have been stolen, and until they're found, the Galactic Games cannot begin. It is up to you as the Tetra Questrian to travel to the dangerous TetraDome in search of the six tablets that have been broken into 64 pieces and hidden throughout the 384 sections of the Tetroid's world. I guess that'll teach us all not to invite the Tetroids to the Galactic Games, huh?

*Tetra Quest* is another example of how an old concept can be turned into something new and special if the right person sets his mind to the task. When all is said and done, *Tetra Quest* is really nothing more than a maze game. You have to find your way from one screen to the next in order to get four coins (four pieces of the tablets). But because of the interesting way you travel on and between each screen, the search for the coins becomes an absorbing and challenging venture.

Each level of the game is made up of a series of screens that are interconnected. Some sections of the screens may be reached simply by traveling on the "limbs," a complicated gridwork. Other sections of the screens may be reached only by using gate switches, teleports or by changing yourself into a phoenix and flying to the area you want to reach. As a phoenix, your travels are no longer limited to the grid.

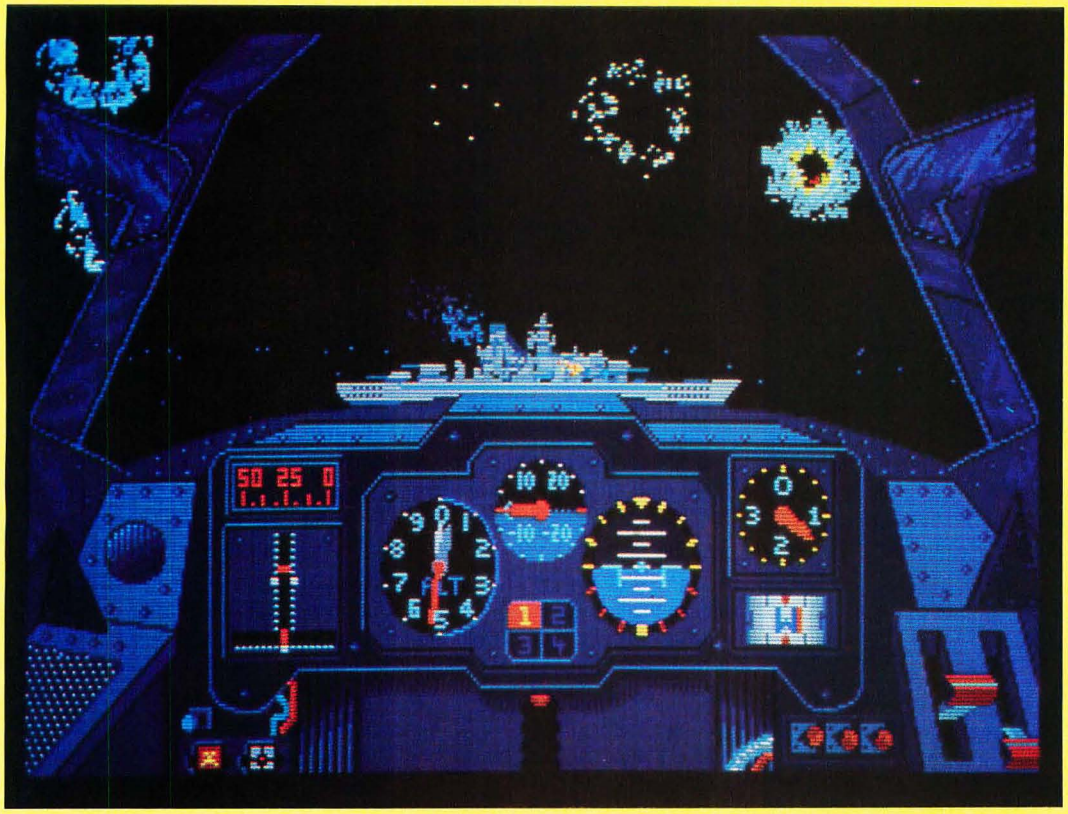
The switches control gates that provide access to different parts of the grid. In order to get to these parts, the gates must be in the right positions. To further complicate the matter, the only way you can know which switch controls which gate is to experiment. And just to make sure you've got a real challenge on your hands, many of the switches control gates on other screens. In other words, you must trigger the switch, and if nothing happens on your current screen, you must make your way through the other screens, trying to see which switch has changed position.

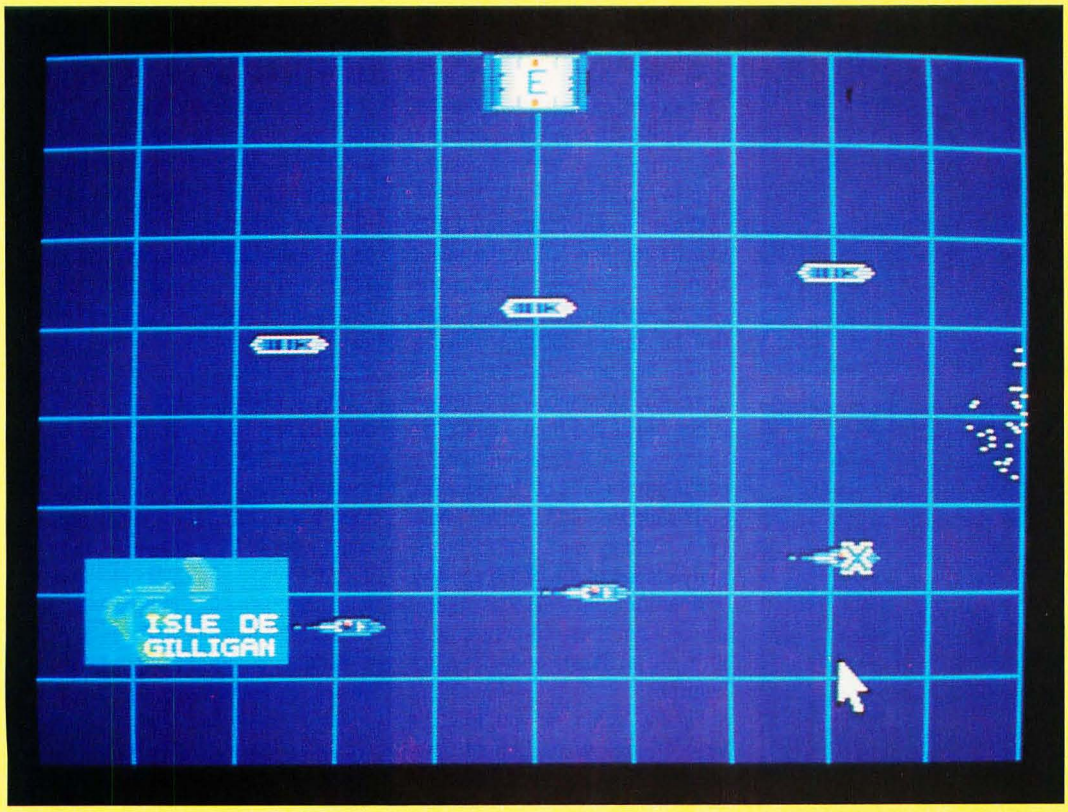
Teleports are easier to use, but can be just as confusing. Once again, the only way to know how they work is to experiment. Moving into a teleport (they look like funnels) causes you to vanish from your current location and reappear somewhere else. You may reappear on the same screen, or you may find yourself several screens away.

The only way to successfully

**continued on page 92**

## Dive Bomber • continued from page 86






**Dive Bomber ▲**

the realism. They actually look like real airplanes drawing closer, not just computer images getting larger. And when you hit one of those planes with your machine-gun fire, you can actually see it burst into flames. You can also see the effects of enemy fire on your own cockpit canopy.

There are other details that enhance the enjoyment of playing *Dive Bomber*. Flip on the ignition switch and the on-screen image shakes as though it were the cockpit of a real fighter plane. And in order to see the controls in the engineer's station, you must flip on the cabin light. But make sure you switch off the light when you go to another station, or you'll be more visible to the enemy and more likely to get shot down.

A mission (game) ends when you're shot down, forced to ditch your plane (because you ran out of fuel or sustained too much damage), or have accomplished your military objective. At the end of a game, an information screen tells how much damage you inflicted on the enemy and how much your plane and aircraft carrier sustained. Points are awarded based on your performance, and the top scores are recorded to disk in a "Pilots' Hall of Fame."

*Dive Bomber* has earned a place in my own ST Entertainment Programs' Hall of Fame. It is one of those rare programs in which a lot of terrific elements blend together to form a whole that's even greater than the sum of its parts. My only gripe is that the two-disk program supports only one drive and therefore necessitates some disk-swapping. In a lesser program that might not be as annoying, but in one as good as *Dive Bomber,* even a few seconds spent swapping disks is frustrating because you'd rather spend that time enjoying the game.

**Recommendation: Buy it.**

*Scott Wasser has been a daily newspaper reporter and editor for the past 12 years and has been interfacing with computers for the last four. He has written columns and feature stories about computer hardware, software and home electronics, and is a regular reviewer for* ST-LOG.

## Elite · continued from page 87

*rotate*) or select your destination and jump into hyperspace for the trip there.

Once you get into space, things get interesting. Your control panel tracks things like shields, fuel laser temperature and other variables. You have a compass for locating planets/stations, and a 3-D scanner that shows objects in space, including whether they are above or below you.

It's also out in space where you have to learn to fight or flee from the pirates (did I mention the pirates?). In the early stages of the game, it's best not to fight, but to instead seek the protection of a space station. The galactic cops are well equipped and do a good job of protecting honest traders close to space stations. Even so, it's best to upgrade your energy banks and lasers (maybe add an energy bomb) as soon as possible. Pirates tend to leave well-equipped traders alone. There are also some hostile aliens out there that shoot at anything that moves.




There are essentially two ways to get ahead in *Elite*. The first is as an honest trader. This is slow but relatively safe. (Yes, you can save the game!) The other way is to deal in contraband goods and shoot up honest traders. The contraband goods are fairly easy to buy, but not so easy to sell, unless you know where to look. Both of these actions, unfortunately, get you blacklisted by the cops; so they may have an unwelcome reception committee waiting for you the next time you dock.

*Elite*'s graphics are incredible. It has simulated, multicolor 3-D, including scrolling star fields, animated with unequaled smoothness. The sequence where the galactic cops came out of the space station to blast me to bits was unbelievable.

Commands are easy to give: You can work with the joystick, mouse or keyboard. The downside is that the controls are set up in an odd way. Imagine a spacecraft that can climb and dive, but can't turn! That's right, moving the mouse left and right (or the joystick, or using the arrow keys), just causes the ship to roll.

While I finally discovered a trick that let me line up on the space station and dock, battles with pirates were exercises in frustration because I couldn't turn my ship to line up the nose-mounted laser. And remember, this review is coming at you from a seasoned space arcader. It turns out that you can adapt the odd control system, and I suppose that I shouldn't have been battling with pirates in the early stages of the game anyway.

Another problem was that I couldn't get the auto-centering to turn off, at least, under joystick control.

Finally, some of you may be wondering how an honest, law-abiding citizen like Dave Plotkin got himself blown away by the galactic cops. Well, it's like this. If you shoot at a space station, the cops get really mad. When using the mouse, you use the left mouse button to fire and the right mouse button to control your speed. Unfortunately, after switching from the joystick to the mouse to try to line up on the space station better, I attempted to slow down by pressing the right button. Yikes! The lasers fired. Ooops, sorry, guys. Guys? Hey, wait a minute, guys. Boom! End of game.

This was repeatable, and so a word of warning: If you use the joystick, don't use the mouse. This bug is unfortunate, since the joystick works better in space battles, but the mouse (sluggish as it is) still works better for the tricky maneuvering around the space station.

There are two key things to remember about *Elite*. First of all, it is a trading simulator, not a combat simulator. Your ship is not designed to outfly or outfight the pirates (did I mention the pirates?), hence the roll vs. turn maneuver. If you can survive until you add some awesome weaponry (like the military lasers—very few people mess with you if you have a couple of these mounted front and rear), you can go marauding. But don't mess with hostiles until you are better equipped, unless you like short games.

The second thing is that *Elite* is not designed to be finished in a short time. One individual I know has been playing for four months. He's got lots of cash, an energy bomb, heavy-duty weaponry, extra everything and has just reached Galaxy 3 (of eight possible galaxies). You could make a career out of playing *Elite*.

Overall, *Elite* is a good game that could be less frustrating if the ship could *turn*. The game has challenge enough without that wrinkle. If you have a low frustration level, you might want to try it before you buy it. But if you live for 3-D graphics, role-playing and lots of arcade action, check this one out.

**Recommendation: Get a demonstration before buying.**

*David Plotkin has been involved with Atari computers for about ten years and is a frequent contributor—of both reviews and software—to* ST-LOG *and its sister publication,* ANALOG.

## Starglider II · continued from page 88

message screen that popped up every time the ICARUS was blown out of the Apogean sky, I failed miserably. I lost count of how many missions ended with the computer mocking me. My repeated failures to save the universe were often greeted with the message: "Try shooting Egrons next time."

Hey, I did try! But most of the time it felt like I was at the wrong end of a shooting gallery. *Starglider II*'s superb graphics, animation and responsiveness told me in no uncertain terms that I was definitely piloting this fast, maneuverable and heavily armed vehicle called ICARUS.

But the constant pounding I took from Egron patrol craft, stompers and even space pirates made me feel as though someone was trying to win a teddy bear by hitting *my* bull's-eye. You know, the old ten-shots-for-a-dollar syndrome.

Admittedly, I was a pretty wretched spacecraft jockey even during those rare moments when nobody was trying to blast me from the sky. I managed to destroy my spacecraft a few times by simply running it into the ground or other inanimate objects. I never quite got the hang of using the ICARUS' grid-coordinates instrument to figure out where I was or how to get where I wanted to go. X and Y axes? Huh?

Compounding my problem was the mind-boggling array of instruments and controls at my disposal. Speed, compass heading and digital clock I could handle. Tri-axial artificial horizons, energy shield status readouts and inductive refuelling were a little out of my realm. *Starglider II* has all the tools required for some slick flying. Unfortunately for Novenia, I find self-service gasoline pumps a technological challenge.

On the other hand, if you've got at least a normal amount of coordination and a little patience, you can probably master *Starglider II* without too much trouble. And you'll have a blast doing it.



The graphics are outstanding: bright, colorful and solid. Personally, I found them to be a little surreal for my taste, which runs more to flesh-and-blood or at least sheet metal and steel objects. But if you fancy yourself a Luke Skywalker-type, you'll get a blast out of the strange creatures and dazzling effects that will cross your path when you're in the ICARUS.

What really impressed me, though, was the sensation of flight conveyed by *Starglider II*. Despite its out-of-this-world premise, the program may be the most realistic flight simulation ever created for the ST. The good graphics—particularly those of the cockpit control panel and the 3-D-like imagery of the planet and objects below—contribute a great deal to that realism.

But the greatest contributor to *Starglider II*'s realism is its animation. Scrolling is so quick and smooth that it's quite easy to forget that the action is taking place on a computer monitor and not in real life. Point the ICARUS' nose down, and you'll swear you're descending in a real aircraft. Increase velocity and climb into the upper atmosphere, and you can almost feel the G-forces tugging at your back.

Response to mouse and keyboard input is no less realistic. All controls work exactly the way they're supposed to. The on-screen images react so quickly and precisely to inputs that it's easy to forget you're operating a computer and not a real spacecraft.

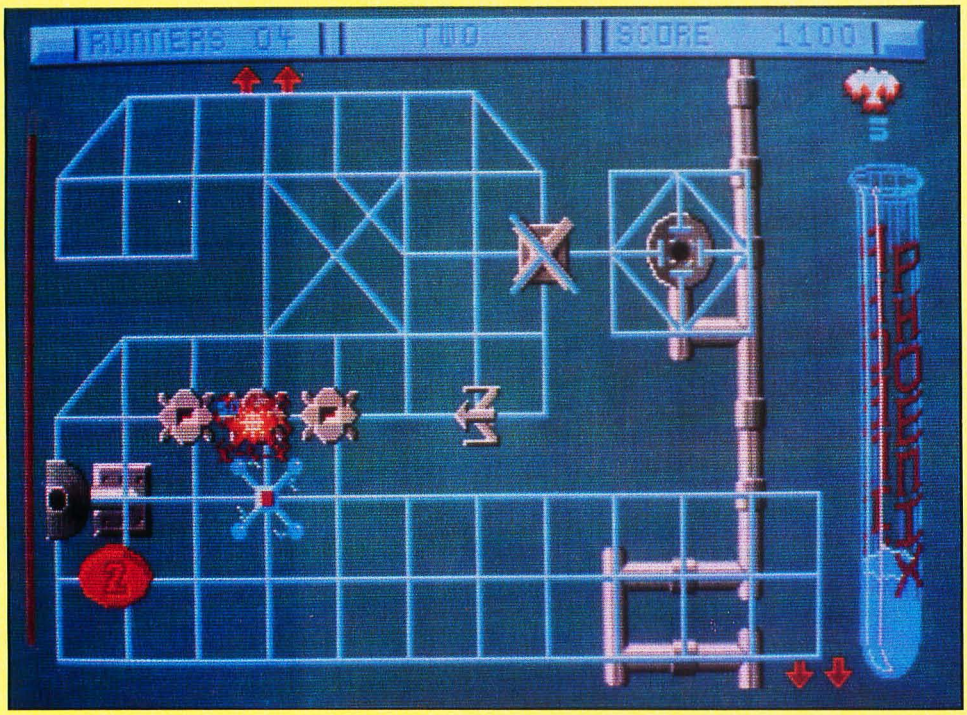
The significance of that cannot be overemphasized because *Starglider II* is in essence an arcade shoot 'em up that demands good eye/hand coordination and quick reflexes. It's sophisticated enough, however, to also require a healthy dose of puzzle-solving, strategic planning and mastery of flight simulation controls.

Apparently, my prowess in those areas left something to be desired. I was putty in the Egrons' hands. *Starglider II* gave me a chance to save part of the universe, but I let the Novenians down.

Maybe you'll do better.

**Recommendation: Get a demonstration before buying.**

## Tetra Quest · continued from page 88




make your way through the maze is to memorize where each of the teleports go and to carefully note which switches control which gates.

Luckily, *Tetra Quest* supplies the player with the ability to transform himself into a phoenix and thus gain the ability to move anywhere on the screen, in complete disregard of the grid. This is a handy way to get somewhere fast, but the power to do this transmutation is limited: You must have the right amount of "phoenix life," the current level of which is shown in a power tube on the right-hand side of the screen. Each time you change to the phoenix, the power goes down.

Most of your time in *Tetra Quest*, however, will not be spent using one of the special transportation devices described above. You will be forced to manuever your Tetra Runner on the grid itself, avoiding the nasty creatures being generated by the "Splitter." These creatures (they come in many varieties) will quickly home in on you just like the ghosts in *Pac-Man*; and if you're to survive to the next level, you must use your guns to make sure that none of them comes near you.

Each level of *Tetra Quest* (there

## Rings of Zilfin · continued from page 87


"Click to continue."

Three-fourths of the screen is filled with your character and his surroundings. The bottom four lines of the screen inform you of special actions or show status messages. These messages include whether your spell casting was successful, as well as the count of arrows in your quiver.

A press of the escape key or selecting Inventory from the menu will bring up a status screen. It lists your possessions, the last place visited and your current direction of travel. While viewing this screen, the game is paused, so you will use it often. It also has a list of every item to be found in the game.

There are a number of items to obtain before your quest is complete. Some may be purchased from town merchants and others are received after performing a special task, defeating an especially devious advisory or visiting a

## Tetra Quest

are supposed to be 96 levels, but I sure never made it that far) is made up of four screens, each of which contains one of the four coins you must obtain. Each of the coins has a number, and you must retrieve the coins in numerical order. This, combined with the gates and teleports, definitely makes for challenging play.

If there is a problem with *Tetra Quest*, it is the joystick response. Just as with most games that force the player to travel a grid-like path, manipulating your movement using the joystick can be frustrating, especially when moving diagonally. The speed at which your Tetra Runner moves increases the difficulty even further. It is lightning fast, and you can make it from one side of the screen to the other in a blink of the eye. If you find yourself traveling the wrong limb, it's difficult to change your direction before disaster strikes.

All things considered, *Tetra Quest*, with its nicely drawn graphics, interesting sound effects and engaging game play, belongs in every ST gamer's collection. You won't be disappointed.

**Recommendation: Buy it.**

***Tetra Quest,* with its nicely drawn graphics, interesting sound effects and engaging game play, belongs in every ST gamer's collection.**

## Rings of Zilfin

hidden location. This makes mapping a necessity. The view map from the menu does not show the location of towns or special areas. I traced the map from the manual and marked towns and special areas as I came upon them.

One helpful feature of the game is the adjustable level of play. I played at Level 9, the easiest. The game play itself is not changed for each level; instead you are given more supplies to start with in the higher levels. This feature gives a novice, like myself, the chance to get out of the gate without getting slaughtered.

The manual is well written and quite informative. I read it three or four times during various stages of the game to glean new tidbits of information. The hint sheet came in handy at the start of the game without giving away too much.

Disk access is acceptable and only occurs when entering and exiting towns. Disk swapping is minimal. You boot with the first disk and replace it with Disk 2 before the actual game begins. Disk 2 holds the data for the first two-thirds of the game and is used for games saves. Only one game can be saved on the disk, but you can make as many copies of the disk as you need. Disk 3 is requested for the final third of the game and will not be swapped unless you save the game or re-enter one of the earlier towns in the game.

Obvious bugs were few, but the game locked up twice under similar circumstances. I would save the game to disk and continue with play by talking to a passerby—when the bombs appeared. Since this always occurred after a game save, I do not consider it a serious bug.

I enjoyed *Rings of Zilfin* so much that I spent three vacation days finishing the game. I would guess it took 20 to 30 hours to complete. I especially recommend this game to those getting started on the road to adventuring because of its logical menu setup, logical and multiple goal obtaining and well-done graphics.

**Recommendation: For adventure fans.**

*Kevin Peck works in the world of MS-DOS by day and plays on his one-meg 520ST at night. He is considering the purchase of a hard disk for his ST so he can get into "serious" ST programming in the near future.*

# SCSI Adapter Boards

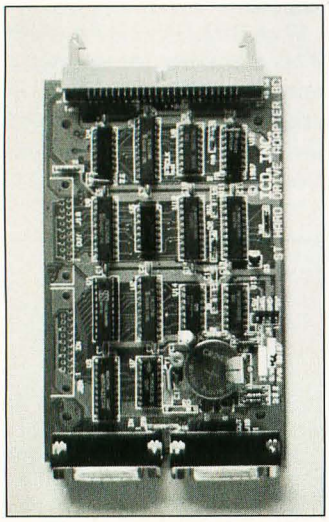
## ST Host Adapter

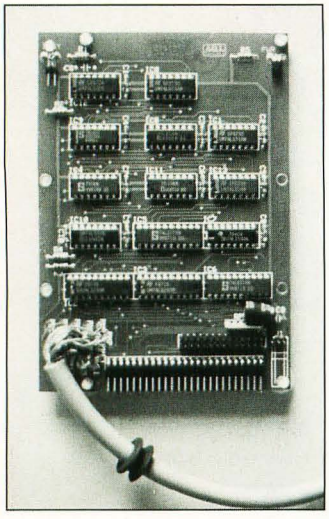
**ICD, Inc.**
**1220 Rock Street**
**Rockford, IL 61101-1437**
**(815) 968-2228**
**$135.95**

## Hard Disk Interface

**Supra Corporation**
**1133 Commercial Way**
**Albany, OR 97321**
**(503) 967-9075**
**$99.95**

**Reviewed by Charles Bachand**





Oh, why did they do it? Why did Atari implement a high-speed DMA port instead of what everybody wants and needs, namely a true SCSI port. It would have only cost them an extra dollar or two to have incorporated SCSI (small computer systems interface) into the original design of the ST computers; but they probably didn't envision back then the large percentage of users who would be adding hard-disk drives to their systems. With more and more STs being used for business application, a hard-disk drive is now considered a must-have item.

The Atari ST's DMA port "can" communicate with SCSI devices (hard disks, streamed tape backup, CD-ROM drives, etc.), but it is not easy. SCSI is not compatible with the DMA port that Atari implemented. The signal lines (SCSI uses a 50-conductor flat ribbon cable, while the DMA uses only 16 signal wires), timing and protocol are different. Some of these differences are slight, but enough to matter—thus the need for a DMA-to-SCSI adapter board.

Several companies that manufacture hard-disk drives for the Atari have designed SCSI adapter boards for their own use. These companies, likewise, sell them separately to users who are hardware hackers who build their own equipment whenever technically and economically feasible. I like to consider myself as being a member of this elite group. (I'm always the first on the scene with a screwdriver in hand, ready to examine the inner workings of the latest piece of equipment to come through the door.)

My approach to building a hard disk was a little different from Andy Eddy's (see "Frankendrive," ST-LOG #21), in that I chose to go with a drive mechanism with an imbedded SCSI Hard Disk Controller. Normally, imbedded SCSI drives are more expensive than going the separate drive/controller route, but you see, I got this great deal on a 54-megabyte unit at a local computer flea market. Well. . .you know.

### Strange as it ever was

You should not confuse a SCSI Controller with a SCSI Adapter. A controller is a dedicated microcomputer that accepts commands and data from your computer system to read/write to a hard-disk drive unit. It then sends the results of your computer's commands (the data being read, acknowledgement codes, error messages, etc.) back to your computer. The SCSI Adapter merely changes the interface signals so that they're now compatible. The data, however, remains the same.

### You're in the army now!

In a sense, the whole thing is akin to the chain of command in the army. The general (your computer) wants a file that he left in his jeep (a track on the hard disk) in the parking lot (hard-disk platter). He gives an order to his aide (the DMA port) to retrieve the information; but since the general's aide can't leave the office, he calls up the captain (SCSI Controller) via the telephone (DMA-to-SCSI Adapter) to relate the general's (computer's) orders.

The captain, who loves to be a captain, yells at the sergeant (the hard-disk controller—yes, there is a microcomputer in each and every drive mechanism as well), telling him which parking lot and which jeep to locate and just what to do with it when he finds it. This sergeant, however, is not into physical labor, so he yells at a lowly private (the hard disk's read/write head), who just happens to be already running around the parking lot, and tells him which jeep (track) to go to.

Unfortunately, the lowly private finds much more than one folder (sector) in the jeep. These jeeps seem to hold 17 folders—although

some camp parking lot's vehicles will hold 25 (RLL encoded drives). The private has no choice but to take all 17 folders back to the sergeant, who can't make heads or tails of them either, and passes the whole mess along to the captain, who, being a relatively smart guy picks out the correct folder and sends it along with a report (command acknowledged, etc.), back up through the proper channels (SCSI/DMA Adapter to DMA port to your computer).

## Faster than a speeding bullet

Of course, all this talking back-and-forth happens very quickly, taking less than a second to read or write a 100K file. You have to know what you are doing when designing hardware that deals with binary information at these speeds. Two companies that certainly have their act together in this regard are ICD and Supra—both of which manufacture and market Atari hard-disk systems, as well as the all-important DMA-to-SCSI Adapters. We'll take a look at their adapter boards in this review.

## Hey, buddy, got the time?

The ICD SCSI adapter board is a multifunction unit. Not only does it handle the all-important functions of communicating with your disk drive, it also keeps continuous track of the current date and time. This is accomplished through the addition of a few extra parts to the ICD PC-board, namely, a real-time clock chip, assorted transistors, resistors and capacitors, a quartz timing crystal and a lithium battery. The battery powers the clock circuitry continuously so that the board never forgets what time it is, and, since it is a lithium battery, you needn't worry about it dying on you for at least four to five years.

The clock communicates with your computer as SCSI Device 6. If for some reason you need this slot number and don't mind losing the clock functions, there are instructions in the manual on how to deactivate it. This involves cutting two traces on the PCB and also adding two jumper wires. Fortunately, the chances that you might need to perform this operation are slight, to say the least.

The DMA I/O ports (two DB-19 connectors on the back of the board) have their traces running to solder pads located on the side of the board also. This was done in case you or ICD needed to change the position of the SCSI Adapter as it is mounted in the drive case to accommodate any future needs. Another advantage to using two DMA connectors (one IN and one OUT) is that you can now daisy chain other DMA devices onto your system. This means that you can also use an Atari or Supra hard-disk drive, or even the Atari laser printer in conjunction with an ICD-equipped drive.

The 50-pin header connector that retains the ribbon cable going to the SCSI Controller uses gold-plated pins to both decrease electrical resistance and reduce the chance of corrosion. The connector also incorporates ejector latches to ease the strain put on the ribbon cable by insertion and removal. The cable, with every other wire being tied to the electrical ground, acts as a shield that blocks radio frequency noise from being transmitted to adjacent conductors. Coupling this with the use of signal buffering chips and noise-reducing terminating resistors allows you to have up to 20 feet of ribbon cable (the design limit for SCSI devices) between the Adapter and SCSI Controller—a sharp contrast to the two- to three-foot maximum limit put on the DMA cable itself.

The ICD hard-disk formatting software will handle a wide range of SCSI Controllers and drive mechanisms—including several imbedded SCSI drives similar to my flea market special. The software can operate on over 120 different drives and over a dozen different controllers. If your unit is not on ICD's list, but you have the specifics, such as number of heads, number of tracks, step rate, etc., concerning it, then it is a simple matter to edit the ICDFMT.DAT file to include your particular unit.

If you happen to have an oddball controller card that you'd like to use and also have access to *Personal Pascal* (previously from OSS but now marketed through ICD), you can write your own device driver for it that the formatting software will recognize. Several examples of controller software are included for Adaptec and Xebec controller boards to get you started.

## Supra, simply Supra!

The adapter board from Supra Corp. is almost as nice as the ICD unit, and both units have many features in common. So I'll only be pointing out the differences here. The PC board is somewhat smaller due to the fact that it does not contain the circuitry to accommodate an on-board real-time clock. A clock chip was one of the first things to be added to my ST way back when, so I did not miss that particular duplication of effort. This would also hold true for the new Mega ST computers that come with a factory-installed clock.

There is no DMA-through connector. In fact, there are no on-board DMA connectors at all to speak of. It seems that the female DB-19 connectors used by Atari for this application are somewhat hard to come by; so Supra decided (at least they did on this version of the board) to solder the three-foot DMA cable directly to the appropriate solder pads.

Of course, there are advantages as well as disadvantages in doing this. There is now no need to cut a large hole in order to mount a connector: A 3/8-inch opening will do nicely to pass the cable through. However, removing the cable from your drive now becomes difficult to nearly impossible since the cable has become a permanent feature of your hard disk.

Supra's formatting software, while not handling as many different drives as ICD's, does allow you to add in data on your own hard disk via editable text fields within the formatting program—somewhat easier than calling up a text editor to change a configuration file as ICD chose to do. The software winner for me was Supra's SUPEDIT.PRG program. This utility allows a *knowledgeable* (I stress that word heavily) person access to the deepest recesses of the hard disk: the nooks and crannies containing the boot sector, partition information and file data. This program has saved me many a time, but for those who do not know what they are doing, SUPEDIT.PRG is like a loaded gun with the safety off! It is very easy to make a complete mess of a perfectly good hard disk. Be warned.

## Which one, which one?

If you are curious as to which board I finally gravitated to and why, you just might be surprised. Both units performed equally well in my application. I did not need the on-board clock or DMA-through connectors, and both the DMA and SCSI cables were limited to two feet in length.

If I had needed to run the SCSI cable to lengths greater than six feet (with ICD's additional noise reduction circuitry we can go up to a maximum of 20 feet here), or if I didn't already own a real-time clock, I think I would have chosen the ICD unit over Supra's almost immediately. But....

## Do as I say, not as I do!

For me, for the most part, it was still a toss-up. Now, since I couldn't very well install both adapters in my hard-disk case—an IBM AT-style tower case—I chose to use the Supra board. The reason? Its mounting holes more closely matched those in my case than ICD's holes did! Yes, it *is* a stupid reason, but then I really hate drilling holes. If things had matched up better the other way, and, all else still being equal, the ICD unit would be gracing my drive. It's just too bad that my computer doesn't have two DMA ports on it. Then I could have the best from both companies! ■

---

*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.*

# APRIL 1989 · DISK LISTING

**The ST-LOG #30 diskette contains 20 files. They are listed below.**

```
   FILENAME.EXT    FILE TYPE     COMMENTS
   ------------    ---------     --------

\CMANSHIP\
   ANIMATE .C      C             SOURCE CODE
   ANIMATE .PRG    RUN FILE      COMPILED VERSION

\ID_EDIT\
   ID_EDIT .PRG    RUN FILE      FONT ID EDITOR
   ID_EDIT .S      ASSEMBLY      SOURCE CODE

\MONOGRAY\
   MONOGRAY.GFA    GFA BASIC     MONOCHROME-GRAY
   MONOGRAY.LST    GFA BASIC     LISTED GFA SOURCE
   DEMO1   .LST    GFA BASIC     DEMO IN LIST FORMAT
   DEMO2   .LST    GFA BASIC     DEMO IN LIST FORMAT

\PAINT\
   MPAINT  .ACC    ACCESSORY     MULTI-PAINT
   MPAINT  .PAS    PASCAL        SOURCE CODE
   COLOR   .RSC    RESOURCE      COLOR RESOURCE FILE
   MONO    .RSC    RESOURCE      MONO RESOURCE FILE

\PDPARADE\
   READTHIS.1ST    TEXT          DOCUMENTATION
   STARTGEM.DOC    TEXT          DOCUMENTATION
   SUPERBT .DOC    TEXT          DOCUMENTATION
   SUPERBT .PRG    RUN FILE      SUPER BOOT
   STARTGEM.PRG    RUN FILE      AUXILIARY PROGRAM
   SUPER_CS.PRG    RUN FILE      AUXILIARY PROGRAM
   SUPER_CS.RSC    RESOURCE      RESOURCE FILE

\STCHECK\
   STCHECK2.BAS    ST BASIC      ST-CHECK

   README  .DOC    TEXT          DISK INSTRUCTIONS
```

DISK INSTRUCTIONS:
Only those files with .PRG, .TOS or .TTP extensions may be run from the GEM Desktop. Other programs may require additional software as shown below.

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 or GFABASRO.PRG |
| .LST | Requires GFA BASIC |

## 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.

# The Disaster Magnet

by Maurice Molyneaux

Have you ever noticed that electronic devices seem to be more prone to weird failures, misalignments and outright disasters than just about anything this side of Ronald Reagan's Cabinet? For example, my faithful Technics cassette deck suddenly went on the fritz the other day. The sound just went flat. The night before, it worked fine. The next morning, it sounded like some deranged audiophile had simultaneously activated Dolby B and C (and maybe even D through Z from the sound of it), and also DBX noise reduction—as well as shoved pillows into my stereo's woofers. There was no gradual degradation of sound; no hint that this was going to happen. Bang! It was just there!

This sudden type of disaster recently happened to my friend's hard drives. He parked and shut off his hard disks (he has two mechanisms), and when he turned them back on the next day, the second drive wouldn't work. It turned out that the drive motor had burned out, and the drive couldn't get up to speed! One night, all is fine. The next day, it's pushing up daisies.

Then there are the cases of floppy disks that go bad, floppy drives that write disks that other drives of the same type can't read (and, eventually, neither can they!) and programs that suddenly begin bombing when they had behaved themselves for a year or more. Or how about those important data files that get jumbled, and you find that the only backup disks you have (if any) have gone dead too?

Sound familiar? If you've used a computer for any amount of time, you've probably run into these problems.

Just why do these things happen, so suddenly, like a disk write out of the blue? Are we slaves to the cruel whims of fate? Are we being punished by the gods above? Could there be scientific explanations? Are there reasons why electronic devices—computers in particular—seem to act like magnets for disaster?

Umm . . . we'll try the scientific explanations.

Let's look at those hardware failures. In the case of both my cassette deck and the freaked-out floppy drive, one explanation that comes to mind is that both began to act up a while ago, but the misadjustments were so slight as to not really affect the performance of the devices. But then, as they slid farther and farther from the "ideal" state, they eventually crossed the threshold from "stable" to "clinically dead." They became stiffs. The curtain came down, and they joined the *bleeding choir invisible.* Bereft of life, they now rest in peace (pieces?). Ahem. Because an electronic device generally either works or it doesn't, there aren't often any stages in between, sad to say.

As for the hard drive that kicked off, a power spike or voltage irregularity (otherwise known as an electron riot) during the drive's power-up could have done it in. Either way, the result is lost data, wasted time and most terrible and frightening of all . . . maybe even a *repair bill!*

How about those dying disks, strangled software and curious crashes? (Please absolve me of my abysmal attempts at alliteration.) Well, the explanations can be one (or more) of many, from the Earth's magnetic field playing tricks to cosmic rays clobbering a crucial bit on disk and/or in memory to the monitor's magnetic field (or Junior playing with the magnets from the refrigerator) mucking up your important data. And let's not forget the ever-popular "the dog chewed it up" excuse.

Personally, I think it's the cosmic rays playing pranks on the disk's electrons. Consequently, the angered electrons form a labor union and go on strike. But, because the cosmic rays never cease, neither does the strike—and your data is effectively lost. This is not mere conjecture. I have data to support this conclusion. Now where did that copy of the *Enquirer* go?

On a related track: It's obvious that your monitor's screen gets dusty due to the fact that static electricity (that's the scientific term for electrons scuffing their feet on the carpet) is a wonderful glue for all those sneaky dust particles that are itching to clog your computer's digital arteries. But, have you ever noticed how *smoke* seems to make a beeline for your computer? Although I don't allow smoking anywhere near my computer or anywhere in my home period, I've noticed this phenomenon elsewhere. Unlike the gradual adhering of dust, there seem to be currents that draw smoke toward your computer. Now, clearly, magnetic fields and static electricity alone cannot account for this, so what's the explanation?

Here's my guess. I call it the "Modified Bradshaw Smoke Convection Theory" (named for my friend, Mark Bradshaw, who came up with the premise as he tried to determine why cigarette smoke always gravitated towards nonsmokers). The theory works as follows: When a person smokes, his body's surface temperature decreases slightly, radiating less heat and thus warming the surrounding air less. This creates a high-pressure zone. The computer, meanwhile, with all its warm little electronic components, generates more heat, causing the surrounding air to warm, rise and expand. This creates a low-pressure zone. Follow me? Okay, so now we have a smoker with a high-pressure, smoke-filled zone around him and a nearby computer with a low-pressure zone around it. A little basic physics (or meteorology) will tell you what happens next. The smoke from the high-pressure zone moves into the low-pressure zone and bingo! Your computer smells like the winner of the Benson & Hedges Smoke and Choke Off.

Either that, or computers, like some people, just enjoy the smoke and suck it up of their own volition, not concerned about the danger to the health of their data. My suspicion is that most electrons are addicted to nicotine, and when they're all charged up by your computer, they start sucking really hard to get at that smoke.

Of course, one of the greatest mysteries of all is how something this absurd gets printed in an otherwise respectable computer magazine in the first place. There's an explanation for that too, known as "Einstein's Theory of Relative Humor," but I'll spare you the details. ▪

ST-LOG *invites all authors to submit essays for possible use in the* Footnotes *column. Submissions should be no longer than 1,500 words and may be on any aspect of Atari computing. Any style or type of essay is acceptable: opinion, humor, personal experience. Creativity is a plus. Submissions should be sent to:* Footnotes, *c/o* ST-LOG, *P.O. Box 1413-M.O., Manchester, CT 06040-1413.*





## 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** MCPWW

☐ **12 Issues with Disk $79** DCPWW

☐ 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 June 30, 1989

**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, voice, 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 #115 ON READER SERVICE CARD.

# Backup For The Future


Before you go any further, back up. And take a good look at what you're leaving behind.

Think about all of that data you've stored away. Information that's taken you months, even years, to enter. Not to mention all of the files you've collected. The programs you've written. The music you've composed.

Now, try to imagine the future without it. All that work, *lost*. All that time, *spent*. All that money, *gone*. And all because you failed to backup for the future.

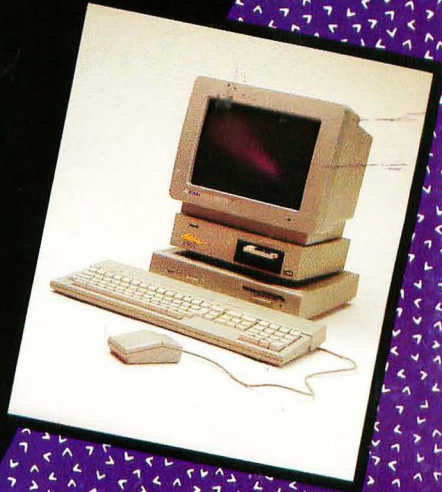
Preparing for your future is what the FA•ST Tape Backup does best. Offering the promises of tomorrow without the risks of today. It's the only tape backup system for the Atari ST that makes data loss a thing of the past— for the good of your future.

So now there's no turning back. Not with the FA•ST Tape Backup, as it secures up to 155 megabytes of valuable data at a rapid rate of six megabytes per minute. It knows exactly how to keep all that data on the right track too, either by file or partition restoration. With back-to-back enhancements that make it effortless.

Like its user friendly GEM-style interface, an image backup mode for Magic Sac or Spectre 128 partitions, a built-in battery backed clock and daisy chaining through its DMA and SCSI ports. And for the times you need to share the FA•ST Tape Backup with other computers, its one-of-a-kind push button SCSI ID switches put the selection at your fingertips.

If that's not enough, the FA•ST Tape Backup can pack a complete hard drive right inside its compact case at your request. Choose from either a 50, 100 or even 200 megabyte system— when you really need a Backup with a lot of drive.

The FA•ST Tape Backup. Think about it... a future without data loss. And then prepare for it. Because you really do have too much to lose.



Call or write for a free catalog today.

ICD, Inc. • 1220 Rock Street • Rockford, Illinois 61101
(815) 968-2228 • BBS: (815) 968-2229



CIRCLE #116 ON READER SERVICE CARD.