SPECIAL • GAMES • ISSUE

ST LOG

THE ATARI ST MONTHLY MAGAZINE

64736

OCTOBER 1988 ISSUE 24

USA $3.50
CANADA $4.75

PRESENTING
MONKEYS AND BALLOONS

EVIEWS
OF 5 REAT GAMES

0 73361 64736 8 10

Refresh Your Memory

And Keep Your Cool.

Introducing the ST Hard Drive System from ICD that refreshes your memory better than any other ST hard drive around. *No problem.*

It's the drive that not only looks cool, but stays cool too. All because of a built-in fan that knows exactly how to beat the heat and maintain a calm, cool and collected environment . . . even in *your* most heated situations. *No sweat.*

And, it's the hard drive that takes a refreshing approach to aesthetic case design as well. See for yourself. It's easy on the space, fitting perfectly under the monitor. And it's easy on the eyes, tailored to look great in the company of your Atari ST. With adjustable front legs, your monitor gets the lift it needs for comfortable viewing. *No strain.*

Despite a sleek and compact exterior, the ICD ST Hard Drive System is packed full of overwhelming enhancements. Like an internal clock that tags each file with up-to-the-minute time and date information. Not to mention expansion capabilities that welcome the connection of up to six SCSI devices and daisy-chaining Atari's DMA Bus (ACSI). It's available in more memory capacities than you can imagine. With storage ranging from 20 megabyte systems up to 280 megabytes. And, there's dual drives too, that double your protection and double your confidence. *No stress.*

So, the next time you think about a hard drive for your Atari ST, think about the countless ways we can refresh your memory. It's the only drive worth remembering. Because it's from ICD. *No wonder.*

For further product information, please call or write for our catalog today.

ICD

1220 Rock Street
Rockford, IL 61101-1437
(815)968-2228
MODEM: (815)968-2229
FAX: (815)968-6888

Atari ST is a trademark of Atari Corporation.

CIRCLE #101 ON READER SERVICE CARD.

EDITORIAL

by CLAYTON WALNUM

Although there has been no official word, there is little doubt that Atari will soon release a 68000-based game machine. Most owners of ST computers greet this news with less than enthusiasm because they are afraid that having a game machine based on the same chip as the ST will further decrease the ST's credibility in the computer marketplace—and they are also afraid that Atari will name the new entertainment product the *ST Game System*, thus placing the "final nail in the ST's coffin."

Unofficial word from Atari is that, should the 68000-based game machine become a reality, it will do everything possible to disassociate the machine from the ST computers. "But," cry the worried ST owners, "look what Atari named their last entry into the entertainment marketplace: the XE Game System! They sure didn't try to disassociate that machine from its 8-bit computer counterpart!"

And, of course, that's a good point. However, the situation in the 8-bit computer market is very different from that of the ST market. From the viewpoint of program and hardware developers, the 8-bit computer is dead and buried. Most of the developers have moved on to other machines, have climbed the technological ladder hand-in-hand with the hardware and are seeking their futures where the future lies. Atari's philosophy in releasing the XE Game System was to not only have a new, high-end video-game machine on the market to provide some competition for the now incredibly successful Nintendo, but also to try and stir up some interest in the 8-bit computer line by giving developers the ability to hit two market segments with one product.

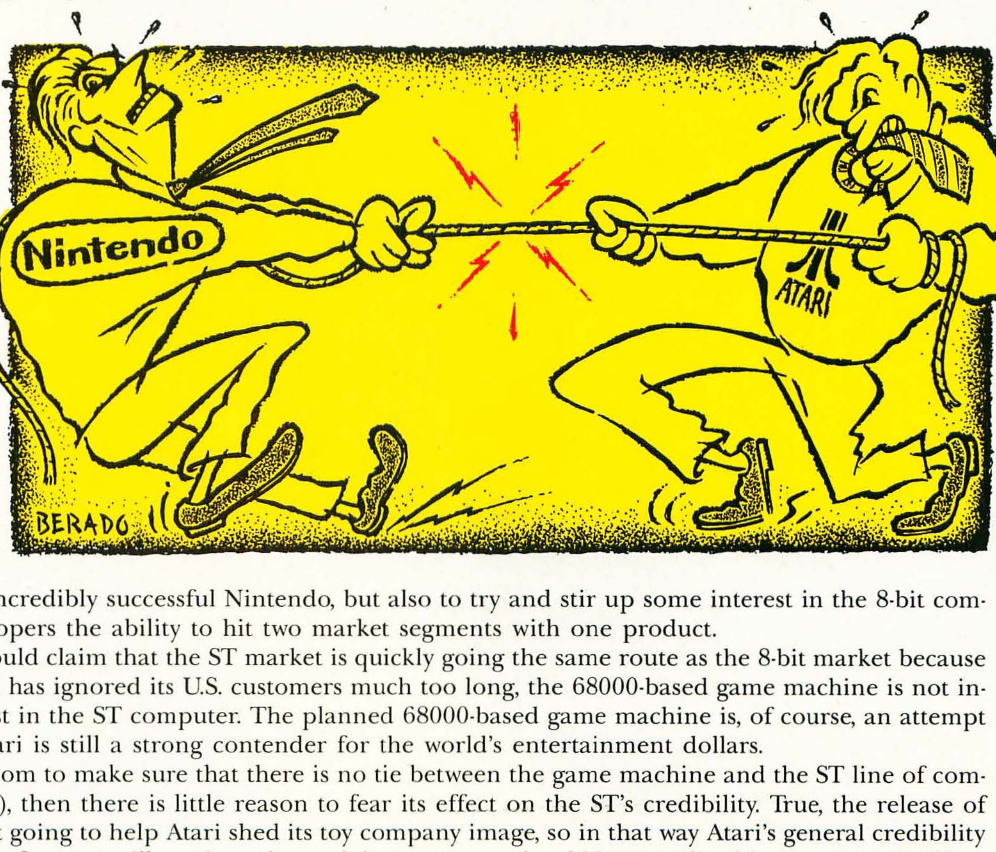


Illustration by John Berado

Though some people would claim that the ST market is quickly going the same route as the 8-bit market because (among other things) Atari has ignored its U.S. customers much too long, the 68000-based game machine is not intended to stimulate interest in the ST computer. The planned 68000-based game machine is, of course, an attempt to show Nintendo that Atari is still a strong contender for the world's entertainment dollars.

If Atari has enough wisdom to make sure that there is no tie between the game machine and the ST line of computers (and they surely do), then there is little reason to fear its effect on the ST's credibility. True, the release of a new game machine is not going to help Atari shed its toy company image, so in that way Atari's general credibility as a serious computer manufacturer will not be enhanced, but everyone should have realized by now that Atari has no intention of leaving its toy roots behind. They are out to grab the dollars where they can get them, and, for Atari, one of those places has always been the video-game market.

It will be interesting to see what Atari comes up with for its 68000-based game machine. It'll be even more interesting to see how they will handle its marketing. Will they do as they've always done and trust word-of-mouth to sell their new machine? Or will they adopt a more aggressive strategy and invest some advertising dollars?

Either way, Atari will be facing some difficulties. Without advertising, the new game machine will never provide much competition for Nintendo. On the other hand, if ST owners turn on their television sets and see ads for the new game machine, there will be a high price on Jack Tramiel's head. "Where," they will insist, "are the ads for the ST?"

IN THIS ISSUE

ST-Log (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1988 L.F.P., Inc. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. All rights reserved on entire contents; nothing may be reproduced in whole or in part without written permission of the publisher. U.S. subscription $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscriptions: add $7 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.

STAFF

Publisher: Lee H. Pappas. **Executive Editor:** Clayton Walnum. **Art Director:** Andy Dean. **Managing Editor:** Dean Brierly. **East Coast Editor:** Arthur Leyenberger. **West Coast Editor:** Charles F. Johnson. **Contributing Editors:** Ian Chadwick, Frank Cohen, Maurice Molyneaux, Steve Panak, Andy Eddy, Douglas Weir. **Copy Chief:** Katrina Veit. **Copy Editors:** Sara Bellum, Anne Denbok, Pat Romero. **Chief Typographer:** Klarissa Curtis. **Typographers:** Judy Villanueva, David Buchanan, Leita Wichman. **Contributors:** Doug Harrison, Jeff Makaiwi, David Plotkin, Bob Retelle, Daniel Silvestri, TG, Karl Wiegers. **Production Director:** Donna Hahner. **Production Assistant:** Steve Hopkins. **Advertising Director:** Paula S. Thornton. **Advertising Production Director:** Janice Rosenblum. **Subscriptions Director:** Irene Gradstein. **Vice President, Sales:** James Gustafson.

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

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

ST LOG
THE ATARI ST MONTHLY MAGAZINE

October 1988
Issue 24

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

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

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

PERMISSIONS

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

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

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

In addition, any programs used must state that they are taken from **ST-Log** Magazine. For further information, contact **ST-Log** at (617) 797-4436.

SUBSCRIPTIONS

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

AUTHORS

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

Cover Illustration by ALLAN HUNTER.

Interior photography: DON CAROLL© Courtesy IMAGE BANK, Los Angeles TWO FUTURES: Courtesy WESTLIGHT, Los Angeles.

ST NEWS

The ST night shirt

For those of you that simply cannot live without the latest accessory for the ST, Kristy Computers is now selling a T-shirt that says "I Love My ST!" Now you can sit beside your ST with a cheerful grin and lightened heart. The T-shirt comes in a bunch of sizes and makes for a great gift for a loved one.

Kristy Computers
P.O. Box 3205
Lorain, OH 44052-7205
(216) 324-2240
CIRCLE #102 ON READER SERVICE CARD.

Bubble Ghost pops onto the ST ▶

A new game from Accolade has been popping up on ST screens. *Bubble Ghost* is a phantom ghost that moves through 35 rooms of a huge mansion, delicately blowing a small soap bubble. Each room is laden with sharp objects, so a certain amount of blowing skill is required. The game was developed by Infogrammes, a French ST developer and offers good game play and lots of interesting graphics.

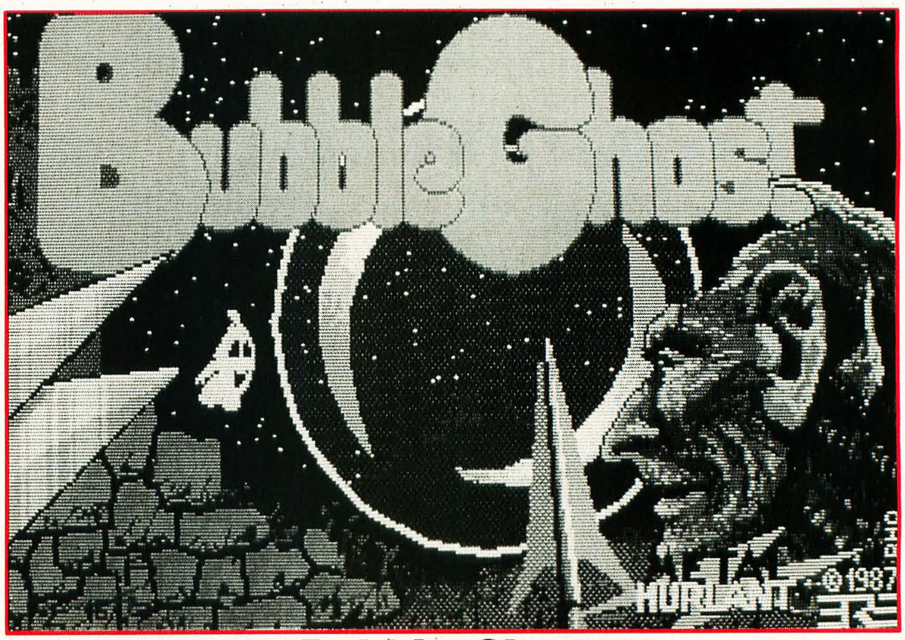
Accolade
20813 Stevens Creek Blvd.
Cupertino, CA 95014
(408) 296-8400
CIRCLE #103 ON READER SERVICE CARD.

Scarry stuff...kids!

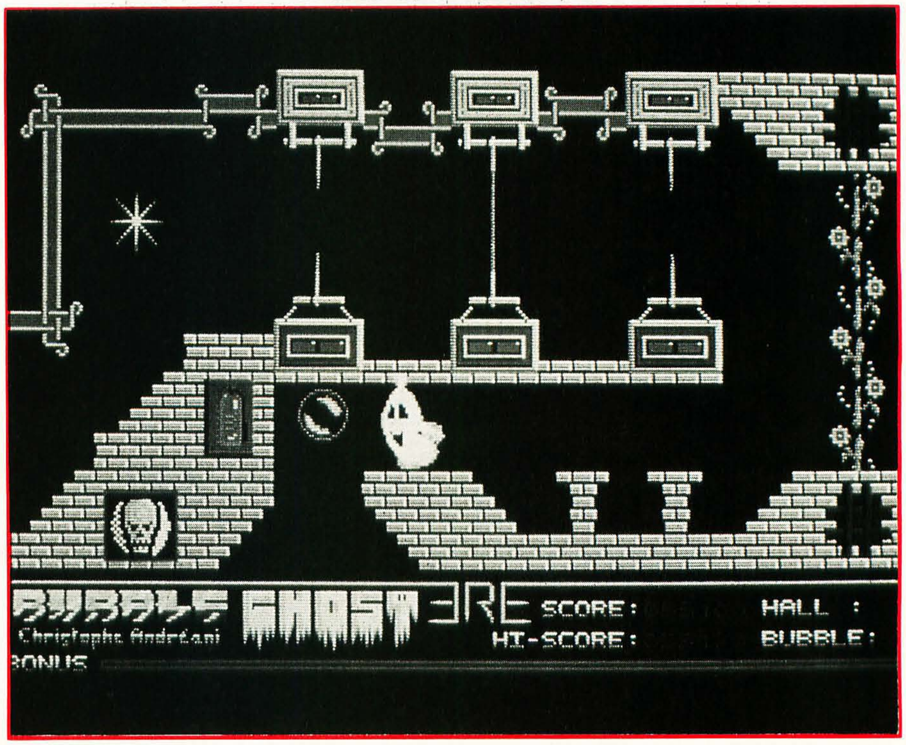
This time on *Monster Chiller Theatre* we have a terrific new show called *Vampire's Empire.* Journey with Dr. Van Helsing to the lair of Dracula himself! On the way, Van Helsing lays a trail of mirrors which bounce sunlight all the way down to Dracula's crypt. Half-human creatures, vampires and other assorted creatures hinder your success.

Vampire's Empire offers some of the better graphics for the ST. Characters walk and float through the 200+ screens with an eerie glow. A standard Atari joystick is all that's required to have some fun.

DigiTek, Inc.
10415 N. Florida Ave., Suite 410
Tampa, FL 33612
(813) 933-8023
CIRCLE #104 ON READER SERVICE CARD.



Bubble Ghost ▼



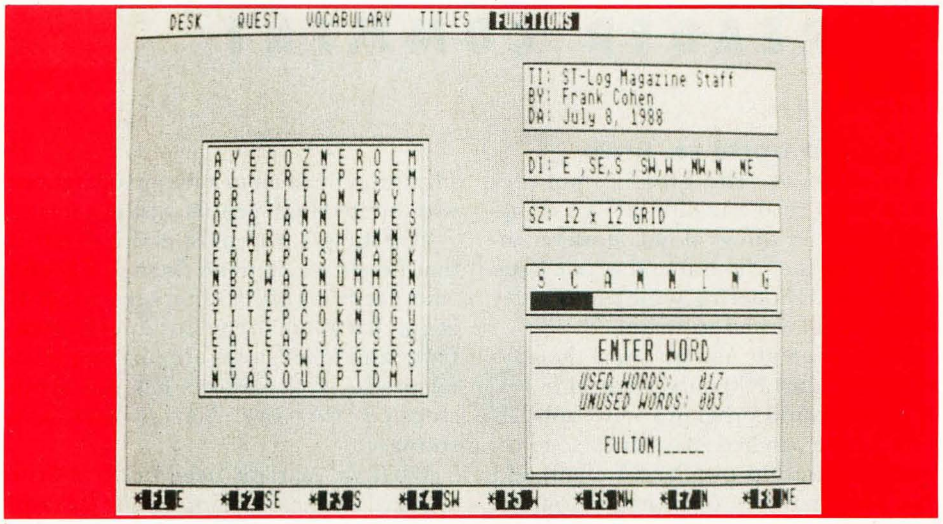
Word Quest ST

Artisan Software has released *Word Quest*, a hidden-word-search game for the ST. Word-search games hide words, phrases, names and other text in a matrix of jumbled letters. Word Quest makes entry and creation of word search puzzles easy and fun, since the program intelligently places any words you enter into the puzzle.

ST-Log used Word Quest to develop the puzzle shown here. In it you will find most of the names of the editors, writers and contributors to ST-Log. There are 18 names in the puzzle, and it took less than five minutes to build!

The program was written in *GFA BASIC*, but it is surprisingly well behaved and nicely GEM-based. For only $17.95, Word Quest makes an interesting addition to your software library. The product is available only from Artisan Software directly.

Artisan Software
P.O. Box 3213
Fontana, CA 92334
CIRCLE #105 ON READER SERVICE CARD.



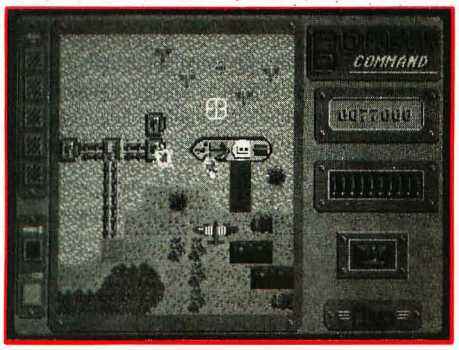
```
A Y E E O Z N E R O L M        TI: ST-Log Magazine Staff
P L F E R E I P E S E M        BY: Frank Cohen
B R I L L I A N T K Y I        DA: July 8, 1988
O E A T A N N L F P E S
D I W R A C O H E N N Y        DI: E ,SE,S ,SW,W ,NW,N ,NE
E R T K P G S K N A B K
N B S W A L N U M M E N        SZ: 12 x 12 GRID
S P P I P O H L Q O R A
T I T E P C O K N O G U       S C A N N I N G
E A L E A P J C C S E S
I E I I S W I E G E R S        ENTER WORD
N Y A S O U O P T D M I        USED WORDS:   017
                               UNUSED WORDS: 003
                               FULTON|_____

F1 E   F2 SE   F3 S   F4 SW   F5 W   F6 NW   F7 N   F8 NE
```

```
A Y E E O Z N E R O L M                       "ST-Log Magazine Staff"
P L F E R E I P E S E M                            By Frank Cohen
B R I L L I A N T K Y I                             July 8, 1988
O E A T A N N L F P E S
D I W R A C O H E N N Y             WORD QUEST     (c)1988  ARTISAN SOFTWARE
E R F K P G S K N A B K
N B U W A L N U M M E N       BANKS                JOHNSON              PANAK
S P L I P O H L Q O R A       BODENSTEIN           KNAUSS               PAPPAS
T I T E P C O K N O G U       BRIERLY              LEYENBERGER          PECK
E A O E A P J C C S E S       BRILLIANT            LFP                  STLOG
I E N I S W I E G E R S       COHEN                LORENZ               WALNUM
N Y A S O U O P T D M I       FULTON               MOOSE                WIEGERS
```

The Martians are Invading!

Bomber Command is a new action adventure game for the ST that puts you in control of a twin engine bomber. Below your flying fortress are freight trains, tanks, trucks, ships and lots of other stuff to blow up! An Atari joystick controls your movements and firepower.



The terrain below the bomber scrolls through woods, oceans and military installations. The game offers nice sound and graphics and is reminiscent of Word War II bombing missions. *Bomber Command* has a list price of $39.95 and is now available through dealers and directly from Mars Software.

Mars Software
P.O. Box 70947
Pasadena, CA 91107
(800) 541-0900 (orders only)
CIRCLE #106 ON READER SERVICE CARD.

Not so quiet on the Amiga front

Irving Gould, CEO of Commodore Business Machines, has announced the Amiga 2500 and Amiga 3000. Prototypes of the new machines were shown at the CeBit show in Hannover, West Germany last summer. The Amiga 2500 comes with a 68020 CPU, while the Amiga 3000 comes with a 68030 CPU. Some new LSI custom chips are also being developed which will further enhance memory management, disk I/O and graphics. Gould did not mention any new prices, but he expects the machines to be out sometime this year.

Commodore Business Machines
1200 Wilson Drive
West Chester, PA 19380
(215) 431-9100
CIRCLE #107 ON READER SERVICE CARD.

Keyboard stiffeners

The first-time ST user commonly says, "I like it! But, the keyboard feels *mooshey*." A new ST accessory solves the mashed potatoes problem with the 520ST and 1040ST. *Firmpak ST* is a $9.95 addition to your ST keyboard that adds specially engineered springs to your keyboard. The new feel is nice and firm; Firmpak ST gives your ST a more professional feel. Installation is quick and easy, no special tools are required. Firmpak ST is now being sold through all the major ST dealers.

Intellect Systems
P.O. Box 119
Atwood, CA 92601
(714) 777-3224
CIRCLE #108 ON READER SERVICE CARD.

Atari delays developers conference

Atari Corp. announced that its planned Atari Developers Conference, which was slated for September 15 and 16, has been put on hold. The conference was to have brought U.S. software developers to Sunnyvale for two days of training and instruction for Atari's ST and Mega computer. According to an Atari official, the conference was to have offered developers a look inside Atari in the hopes that a better dialog between Atari and software developers could be achieved. No date has been set, but Atari Corp. expects the Developers Conference to be held this fall.

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
CIRCLE #109 ON READER SERVICE CARD.

Hewlett-Packard DeskJet support

Migraph has introduced several new GDOS printer drivers for the ST and Mega computers. The new printer drivers allow you to use any GDOS compatible application with plotters, inkjet printers and printers. The new GDOS drivers support: HP DeskJet, HP 747x-7550 Plotter, HPGL compatibles and IBM 7372, Roland Plotter Driver for DXY 880, 885, 980, 985, and Houston Instruments Plotter for DMPL compatibles. Each new driver costs $49.95.

Migraph
720 S. 333rd #201
Federal Way, WA 98003
(800) 223-3729
CIRCLE #110 ON READER SERVICE CARD.

READER COMMENT

The world vs. Atari

I tried not to write. I find it's not very often that I read an editorial in a magazine and utter things aloud. However, after reading the editorial in the June issue of ST-Log and uttering some pretty nasty things aloud, I had to write.

I have been an Atarian since the days when the Atari 800 only came with 16K of memory and programs were only 4K long. I have watched Atari struggle to try and come into its own, and I might add that I became very discouraged towards the end of '85. With the introduction of the ST in '85, I again gave Atari a chance. In the summer of '87, with the introduction of the Mega 4, I again gave Atari another chance. I am no stranger to the history of Atari.

Atari has some serious flaws which it had better correct if it is ever going to make a dent in the American market. Atarians who don't help in the struggle are partly at fault. Don't sit back and say, "It's a small company; give it time." Atari needs to improve the following:

One, Atari must take on a more serious and mature role when dealing with the press. Let me give you an example. At COMDEX in the past, Atari announced that they were going to come out with a $3,000 desktop-publishing system. When the system finally hit the market almost a year later than the date set by Atari, the Mega 4 ST was $2,500 and the laser printer was $1,900—a printer that can only be used with the Atari. At present I am running a HP Deskjet at 300 dpi with a price of $689. There are no bugs in the software, and I can sell it to any other computer user if I have to.

Atari has repeated this again with the CD-ROM. It was announced to be out in February '88—to date, still not seen. Critics of Atari in the non-Atari computer media have become deaf to Atari's announcements because none of the announcements are true.

Two, what is Atari doing about visibility via the TV and movie media?

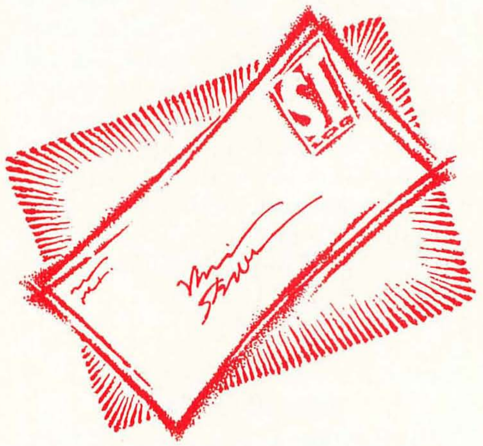
Three, Atari should make as much information available to the user as possible. Have any of you logged onto the Atari BBS in the past? The same news stays on the board until it becomes history.

Four, customer support must improve, as well as the image of the stores selling Atari computers. Besides Federated, most of the stores selling Atari computers are small storefront mom-and-pop operations with limited high-profile business image.

The thing I hate most about Ataris is that I can't recommend them to a friend who needs an office computer. Support is not that good. In the Los Angeles area, fewer and fewer stores are carrying related products. B. Dalton's has just about stopped carrying Atari books and software.

What we Atarians need to do is write to Atari, again and again. Maybe Jack will get the message: Happy users sell more computers. —Gilbert Bush
Alta Loma, CA

I think you missed the point of the editorial. Nowhere in it did I state that Atari was perfect, that they haven't made mistakes in the past and are not likely to make them in the future. We are all aware of Atari's errors (at least, what we call *errors; Atari may have a very different viewpoint of its actions), but that still doesn't change the fact that there is a great difference between* constructive *criticism and* destructive *criticism.*



By all means, write to Atari! Tell them how and why you're dissatisfied and what they can do to make you happier. The point is that, if you're not very careful, you can go full circle with unreasonable griping. Many of the biggest complainers say that yelling and screaming is the only way they have left to make Atari notice them. What they may not be aware of is that that same yelling and screaming is likely to chase away just as many customers as Atari's policies do. If you read the rest of the June issue, then you also saw Mike Donahue's article, "The Absent Revolution." We at ST-Log consider that article to be constructive *criticism because it makes its point without resorting to nasty accusations, name-calling and rumor-spreading.*

And I think that many of the "mom-and-pop" stores you referred to would take exception to your implication that they are somehow inferior. A good many of these private computer dealers are staffed by skilled professionals who know as much (if not more) about the equipment they sell as do the larger dealers. I think it's a sad state of affairs—and a symptom of the disease I was discussing in the editorial—when people think that the only good place to buy a computer is some large chain store. It would be nice to have Atari computers available in the chains in addition *to the private dealers, but there's no need to look down on the smaller stores.*

The self-fulfilling prophecy is a very real phenomenon. People are being turned away from Atari just as quickly by what they hear from other Atari owners as from what they hear from an Apple or Amiga dealer. If that's the way you think things should be, there's nothing I can say. If, however, you want to be part of the cure rather than part of the problem, I would ask that you to at least carefully consider what you say. Make sure that you're not just spreading unfounded rumors, and make sure that you're being realistic.

No matter how you feel about Atari and its policies, chasing people away is not the answer to the problem. —CW

The developer what?

I have heard of two different products for the Atari ST in several different books and magazines, including yours. These products are the *Atari Developer's Kit* and *EMACS*. What are these products, and how can I get them? —Ray L. James
Tampa, FL

The developer's kit is a set of programs with accompanying documentation that is sold by Atari to people who wish to write programs for the ST. The cost of this package is around $300. If you're interested in it, you can get more information by contacting Atari directly at (408) 745-2000. EMACS *(short for* MicroEMACS*) is simply a text editor used mainly for writing program source code. It is available in a public-domain version on many telecommunications services (DELPHI, CompuServe, GEnie, etc.) and from your local user group. However, unless you're planning to write programs for your ST, I doubt you'll find either of these products particularly useful.*

HOWARD MEDICAL COMPUTERS

1690 N. Elston • Chicago, IL 60622 • ORDERS (800) 443-1444 • INQUIRIES AND ORDER STATUS (312) 278-1440

★ 5 STAR FINAL   OCT'88   CLEAR

HMC SAVES BUDGET

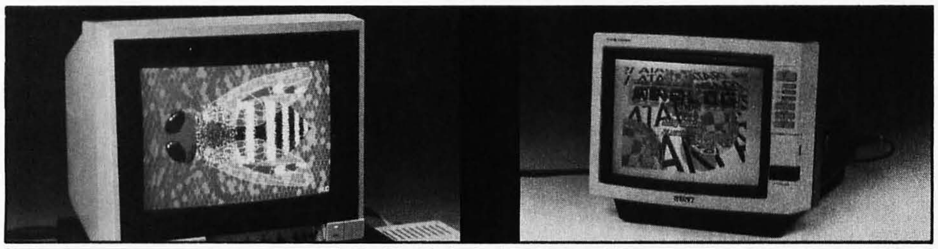
MONITORS

MAGNAVOX color **8CM515** (14 ship) **$^{\$}269$** 3 inputs offer maximum versatility for this 13″ color monitor; RGB Analog for the ST; composite for the 800/XL/XE or VCR and RGB TTL for IBM and Clones. The 640 × 240 resolution with .42 MM dot gives vivid colors and a green screen override for 80 column text. ST RGB cable $19.95 with monitor or $29.50 separate.

MAGNAVOX 8CM 505 $^{\$}220$
13″ Color Monitor 390 × 240 resolution

SONY color **KV1311CR** (15 ship) **$^{\$}499$** The Ultimate 13″ color monitor includes a trinitron TV with remote control; 640 × 240 resolution with a .37 MM dot and a Micro Black screen that is vertically flat for distortion free viewing. Four inputs includes all of those from the Magnavox 515 plus a digital RGB. The colors are incredibly brilliant.

ZENITH green **123A** (7 ship) **$^{\$}67^{50}$** This close out from Zenith features a 12″ non-glare composite Amber screen with 640 × 240 resolution. 90-day warranty valid at our 1200 locations. cable **$19.95**



MODEM

AVATEX 1200 HC (2 ship) **$^{\$}119^{95}$** This Avatex modem is fully Hayes compatible and operates at 300 or 1200 baud. We include express 3.0, a public domain communications program free with each modem.

DRIVE A+ ($^{\$}178^{45}$)
Howards Drive A+ features our DB-1 drive box and our DD-3 MPI double sided double density, 40 track full height drive with case and power supply for full 360K storage.

DB-1 DRIVE BOX ($^{\$}85^{00}$)
Howards drive box hooks up to a ST 1040 to give 2 external drivers. A 3 pole switch will configure an 80 track DSDD (double sided double density) drive or a 40 track DSDD drive or the new 3½ drive. Use with PC Ditto for IBM software compatability.

SIO ($^{\$}64^{45}$)
IDC's PR. Connector gives 2 RS 232 ports for a modem or serial printer and 1 parallel port for a parallel printer like EPSON or STAR. Ask for our special price on cables.

SPARTA DOS COMBO ($^{\$}50^{00}$)
This combo gives you IDC's SPARTA DOS OPERATING SYSTEM and the 2 chips ROM set that allows double sided access for the 1050's.

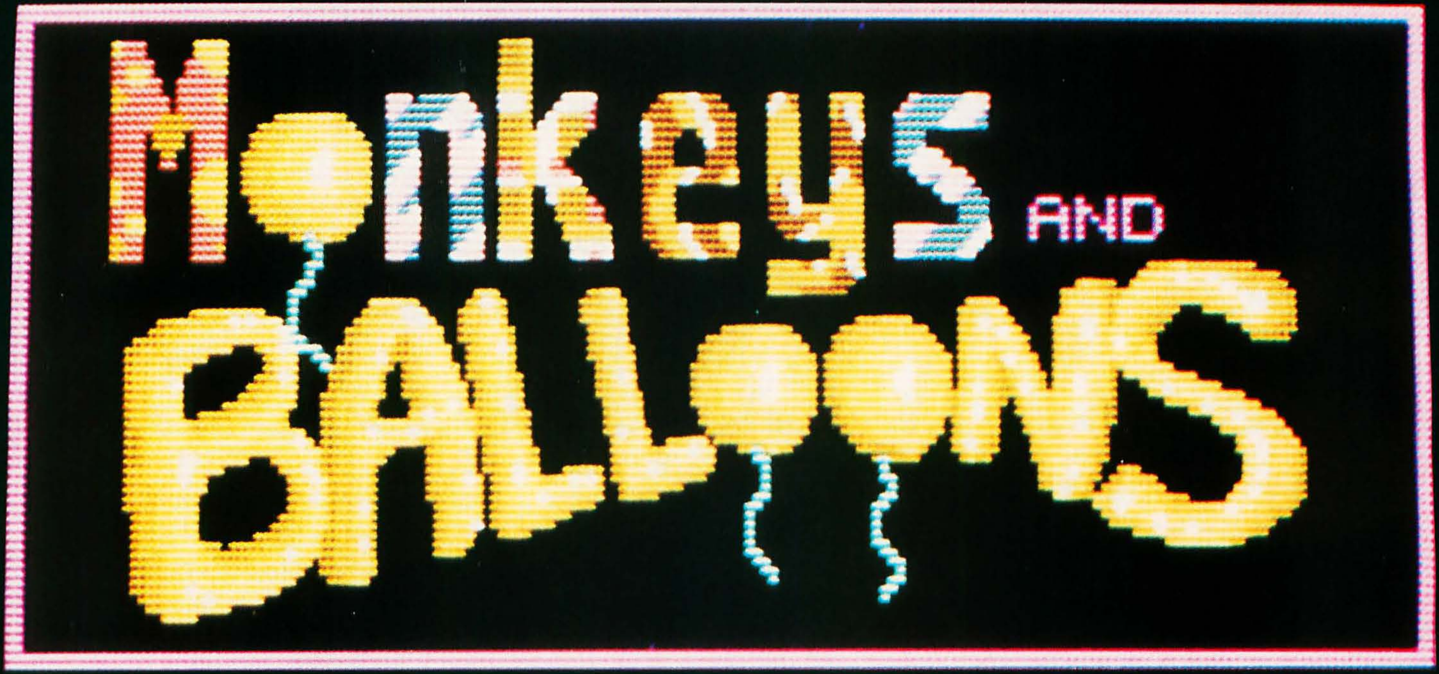
**HMC's Guarantee–
A Promise you can take to the Bank.**

Howard Medical's 30-day guarantee is meant to eliminate the uncertainty of dealing with a company through the mail. Once you receive our hardware, try it out; test it for compatibility. If you're not happy with it for any reason, return it in 30 days and we'll give you your money back (less shipping.) Shipping charges are for 48 states. APO, Canada and Puerto Rico orders are higher.

hotline number

DON'T MISS OUT, ORDER TODAY!
800 / 443-1444
WE ACCEPT VISA • MASTERCARD • AMERICAN EXPRESS • C.O.D. OR CHECKS • SCHOOL P.O.

CIRCLE #112 ON READER SERVICE CARD



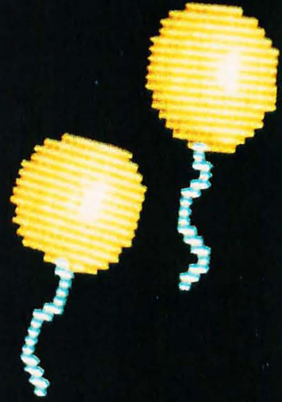
Written by Frank Cohen

Once upon a time, a young college freshman found a company that would pay him the exorbitant salary of $14 an hour to sit in front of an Atari 800 and write a computer game. He didn't know a thing about writing a game, but he was a trained graphic artist and had some experience writing music. So he gave it a whirl. Six months later Datasoft, Inc., released one of its first games, *Clowns and Balloons*.

Since then, the freshman went on to work for other companies, and eventually founded his own computer software company that currently offers ST products. But, the name Clowns and Balloons haunted him as his first, and probably best known, work.

Years later, when his Atari 800 found itself put out to pasture, the idea of writing a new version of Clowns and Balloons kept coming up. The publishers of ST-Log found that, with the right amount of coaxing, Clowns and Balloons would finally see the light of day on the ST computer. A deal was struck and some work begun. But the project was dropped and remained idle for more than a year–while the project was fun to work on, it didn't pay the monthly bills.

When the Clowns and Balloons ST project was recommenced and finished, the name just didn't suit it anymore. The graphics were completely different, the game play more difficult, the music was rewritten and some of the animation was modified. So, the name was changed. Here, then, is *Monkeys and Balloons*.



Required hardware

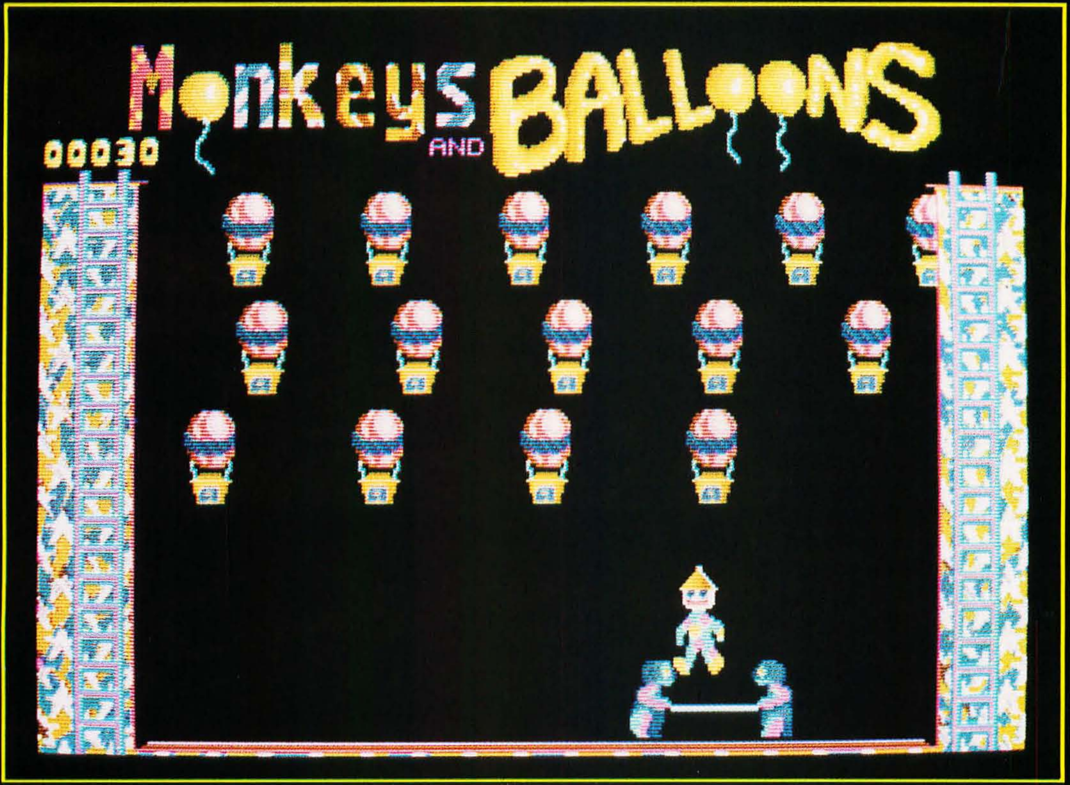
Monkeys and Balloons will run on any Atari ST computer: Atari 520ST, 1040ST, Mega 2 and Mega 4 systems. The game uses your ST's mouse as its joystick, controlling player movement. The program is designed to run in low-resolution mode; so a color monitor is required.

Starting the game

From the GEM Desktop, open the file titled MB.PRG. You can open a file by clicking the program icon once, then selecting the "Open" option from the File drop-down menu. You may also double-click the program icon.

Once the program is started, the title screen will appear and you will hear some music. The music volume is controlled from your color monitor's volume control knob.

After a few moments, a demonstration of the game play will appear. While either viewing the title screen or the demonstration, click the left mouse button to start a new game.



Game play

Welcome to the exciting world of the circus. Cotton candy, peanuts, clowns and balloons are all waiting for your enjoyment. Clowns bounce through the air, while monkeys carefully balance a trampoline close to the ground. Move the trampoline below the clown and everything is fine. The clown bounces higher and higher, trying to catch some of the floating balloons. Miss the clown, and you won't get any bananas tonight!

As the monkey trainer, you control which way the monkeys move the trampoline by moving the ST's mouse left and right. The monkeys can control where the clown will bounce by moving the trampoline slightly when the clown bounces. The clown will bounce straight up and down if he lands right in the center of the trampoline; the farther from the center, the more of an angle he will bounce at.

There are three rows of floating balloons above the monkeys. When the clown's hat touches a balloon, it pops and you are awarded some points. The higher the balloon, the more points are awarded. The object of the game is to clear each row of balloons from bottom to top. If you clear the second or third row of balloons before clearing the bottom row, the higher rows will fill up again with more balloons. When you clear all of the balloons, a bonus award will be given and the next level will begin.

At first, the clown will not bounce very high. The longer the clown continues to bounce, the higher he will go. The clown also bounces faster as the game is played.

While the game is being played, pressing the space bar will terminate the program and return you to the GEM Desktop.

Programming notes

Monkeys and Balloons stores the color palette in use when the program is first started. The program then switches the resolution of the screen to low resolution, storing the current resolution. When exited, Monkeys and Balloons restores the resolution and color palette. This should be standard practice for all games on the ST, but rarely is.

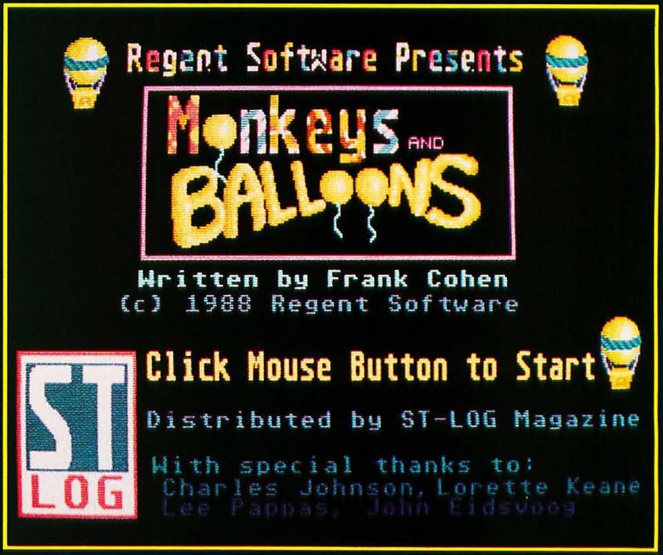
William Robinson (author of *Atariwriter*) has developed his own 68000 assembler, which was of invaluable help in completing this project in a relatively short amount of time. William's assembler takes standard 68000 code and turns it directly into a .PRG program file, ready for running from the GEM Desktop. Other assemblers have to go through two or three steps before they are finished. William's assembler compiles the Monkeys and Balloons source code in under eight seconds! The only problem with William's assembler is that it has not been released to the public. William does not want to support it; so for the moment it isn't available. However, William is easily swayed to public opinion, so writing him a letter might change his opinion.

All of the graphics were developed using *DEGAS Elite*. The graphic files are stored in the normal DEGAS compressed format. When the program is run, all of the graphic files are loaded into the ST's memory and decompressed. The decompression routine took only a few hours to write and can easily be used in other programs. Since the files are DEGAS compatible, you can easily create your own graphics for the game. If you come up with something, be certain to send a copy to ST-Log.

The music routines turned out to be the most difficult part of the project. The note and duration of the music was developed by hand-coding the songs from their original sheet music. Certainly other methods of capturing music data are possible with the ST, but at the time none were known. Once the notes were entered, the problem of how to generate sounds became a problem. Atari's ST documentation is completely void of any information on sound generation. Michtron's *Concise Guide to the Atari ST* showed how to create a frequency, but said nothing about generating notes from a musical scale. Finally, a friend found a small table of frequency to note values in Abacus *ST Graphics and Sound.*

The entire project took less than three weeks of full-time effort to complete. The graphics were developed in about five days, with the balance of the effort spent on programming and debugging.

If you have any ideas or feedback on the game, please send them to ST-Log. There is a whole library of games like Monkeys and Balloons that could be developed with the right encouragement!



Regent Software Presents

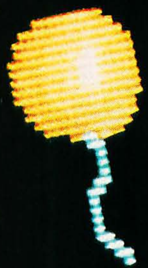
Monkeys AND BALLOONS

Written by Frank Cohen
(c) 1988 Regent Software

Click Mouse Button to Start

Distributed by ST-LOG Magazine

With special thanks to:
Charles Johnson, Lorette Keane
Lee Pappas, John Eidsvoog



Frank Cohen has been writing Atari software since *Clowns and Balloons*, his first 8-bit game. He founded Regent Software in 1985 and has recently completed *Regent Base 2: 4GL SQL Database* for the ST and Mega computers. He can be reached directly on CompuServe, DELPHI and GEnie, or by writing to him at P.O. Box 14628, Long Beach, CA 90803.

The 1988 Summer Consumer Electronics Show

by
Arthur
Leyenberger

Something was different this year. The crowds, the excitement, the exhibitors, Atari: It all seemed *so* different compared to previous years. At first I thought it was just me, but then I realized it was... well...the crowds, the excitement, the exhibitors and Atari. I'll explain.

At the Atari booth there was nary a computer in sight. It was games, games and more games. There were no 8-bit computers, except of course for the XE Game System which, between you and me, is really an 8-bit computer. There were no ST computers except for a couple located in a Hybrid Arts MIDI booth adjacent to the Atari Game pavilion. And, contrary to previous years, there were no 8-bit or ST software/hardware developers located within the Atari booth to demonstrate and hype their products.

Not only was the Atari booth *all* games but so was the rest of the software hall, it seemed. The big exhibitors were Nintendo (rumor was it had some 20,000 square feet) and Sega (its was about half the size of Nintendo's). Funny, the relative size of the Nintendo, Sega and Atari booths more or less reflected their individual percentage of the games market. The only big ST software booth was that of Epyx, which had an array of impressive software for the ST (more on that later).

Overall, however, there were far less computer and software exhibitors than in past years. A long list of familiar names could be made of past CES exhibitors, including Electronic Arts, Mindscape, Batteries Included, Firebird, Infocom, Broderbund, Timeworks, Michtron and on and on. Some of these companies are out of the business. Others either didn't make it to the show (it *is* expensive to have a reasonable-sized booth) or had hospitality suites in conference rooms or at local hotels. Still others combined their resources under the umbrella of their distributor. For example, SoftKat had a booth which contained Broderbund, Electronic Arts, Data East, Timeworks, Activision and Accolade, among others.

Even such product categories as satellite dishes and porn (I mean "adult video entertainment") were notable by their low profile or absence. I suspect that is no great loss to anyone in particular.

The "excitement coefficient" was also lower this year, compared to those in the past. No major software or hardware breakthroughs were announced by anyone. Compared to the early days of the ST and when it was first announced, this summer's CES was almost dull. It also seemed like there was less attendance at the show. Usually, 100,000 people, give or take a couple thousand, roam the aisles and aisles of computers, video, audio and other electronics products. With less exhibitors and smaller crowds, there was less of the frantic, exciting atmosphere in the software hall.

This is not to say that there was no new software being shown for the ST. Just the opposite. There were quite a few new ST titles, although most of it was game software. But that's okay. The ST is a hell of a game-playing machine after all.

What Atari was doing

Almost $1 billion was spent on video games and video-game cartridges in 1987. Atari obviously wants to continue to get a piece of that market, and that's why there is such a big emphasis on video games at Atari, at the expense of the ST and 8-bit computers. In fact, Atari has set its sights on doubling video-game sales for the third year in a row.

Atari has done so well in video-game sales that it has recently joined the ranks of the Fortune 500 companies, with 1987 sales of $493 million dollars. To keep the sales going in 1988, Atari will be introducing more than 19 new games for the XE Game System. Many of these games will be conversions of licensed software. For example, Synapse's *Necromancer*, which appeared on disk for the 8-bit computers four years ago, will be available this year as an XE game cartridge.

New games will also be released for the 7800 and 2600 video-game systems. These games will consist of both licensed games from other manufacturers such as Exidy, Epyx and Lucasfilm, but also new games from, of all people, Nolan Bushnell. Bushnell has signed a video-game-development agreement with Atari in which Axlon will design and develop an unspecified number of video games for the 2600 and 7800 home video-game systems.

FREEDOM STICK
the only infrared joystick compatible with every video game system today, brings new freedom to this popular pastime. No wires, no eyestrain . . . just more fun!

Bushnell, 45, founded Atari in 1972 shortly after introducing the video game *Pong*. Pong signaled the dawn of the video-game generation and helped propel Atari into what later became the fastest-growing company in U.S. history. He sold the company to Warner Communications in 1976 and in 1983 founded Axlon, which develops and manufacturers coin-operated games and designs toys for licensing.

The new products

The most exciting booth containing ST software at CES was without question, that of Epyx. Not only was Epyx a "happening" place to be (it was one of the busiest), it had quite a few excellent new titles for the ST. Several game lines were introduced, including the Action-Strategy series, Masters collection, Sports and U.S. Gold collection.

In one of the Sports games, players use their noodles in the sports parody *Sports-A-Roni.* This zany multi-event challenge takes place in various parts of the world's pasta capital—Italy! Players race down the potholed streets of Naples in the Sack Race, balance a pile of pasta plates near the—you guessed it—Leaning Tower of Pisa, pole-vault the Arno river in the River Jump or climb an olive-oiled pole in Verona's Pole Climb. Additional events include a pillow fight in gondolas, a boot toss amidst the lions in the Colosseumand a pogo-stick party among the Roman ruins.

I know it sounds pretty hokey, but playing Sports-A-Roni is a lot of fun. This is one of the first humorous video games for the ST and a genre not seen since the 2600 game, *Revenge of the Beefsteak Tomatoes.* This multiplayer game will be available by the time you read this and retail for $25.

Another game in the Sports series is *Final Assault*, a mountain-climbing thriller. In this game, your skill will be tested on some of the world's most treacherous peaks. Step by perilous step you'll inch your way across rock faces, glaciers and crevasses, or hang by a finger over yawning chasms, searching for a toehold.

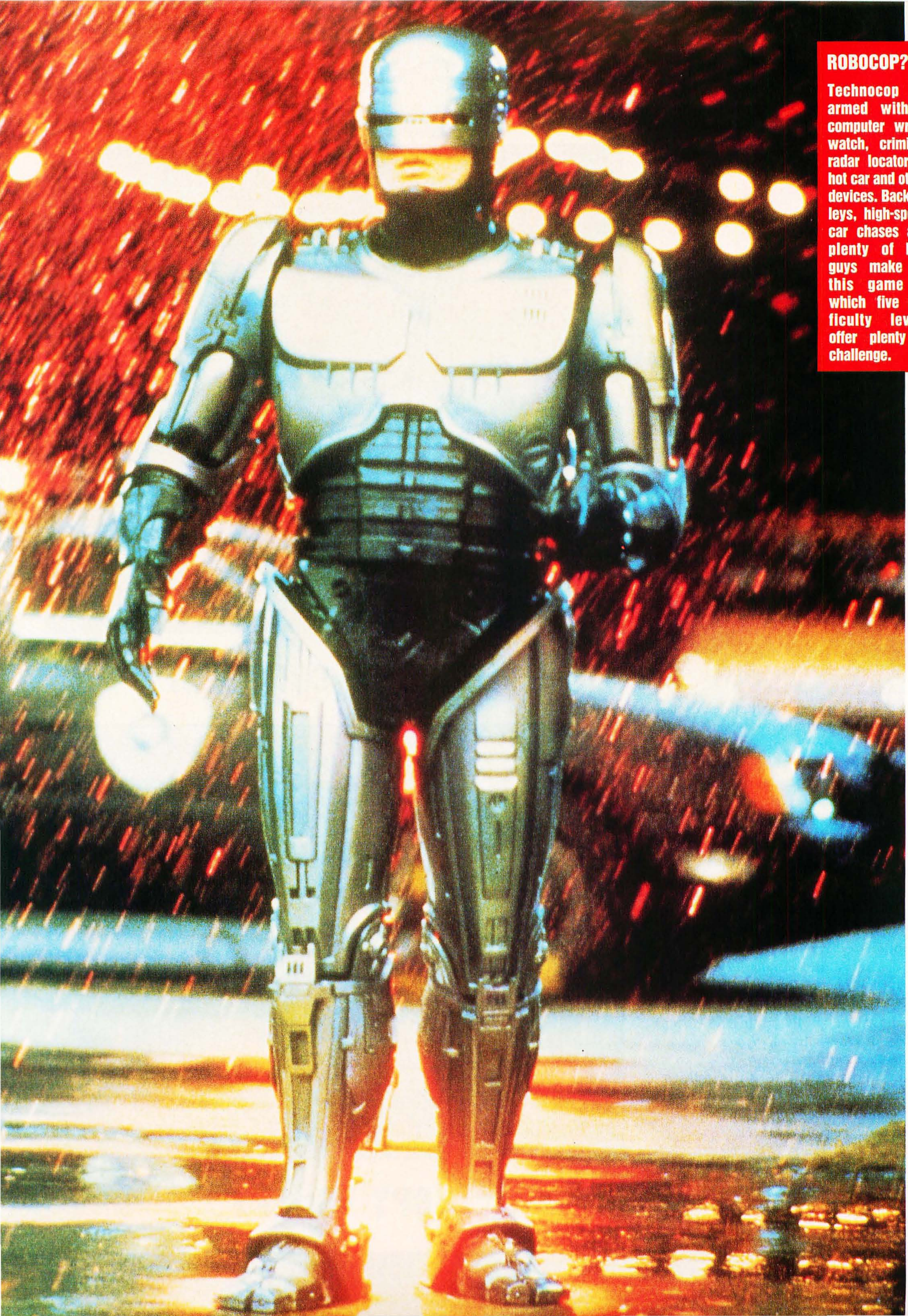
Players may choose to practice their knowledge and skills over the training course before tackling "the big one." They must prepare well for their ordeal and carefully select the items they'll carry in their rucksacks. There are over 50 items available, everything from climbing gear to food supplies. One wrong move can spell the difference between life and death. On-screen features such as temperature, time, altitude, physical state of the climber and a safety guide will aid the adventurer in his bid for the top.

Final Assault will retail for $40 and be available by the time you read this.

Several games that were shown were created by UBIsoft, a leading French developer and distributor of home computer software. *Trials of Honor* is a role-playing game involving medieval pageantry. Part of the Masters Collection for the advanced gamer, Trials of Honor is an adventure game set in the Dark Ages of Europe. The action begins with a battle for a French kingdom after the much-beloved monarch has been murdered and the assassin has assumed the throne.

As the former heir to the throne, the player must journey across the realm, proving his valor and his right to assume rule. He must emerge triumphant from tests of strength and cunning such as archery, arm wrestling, a game of dice and the slaying of menacing monsters in the labyrinth. Graphics in this game are excellent, and a unique effect is achieved by simultaneously showing both the horseman's route along the countryside as well as a close-up of an animated horseman. Trials of Honor is available now for a retail price of $50.

Another UBIsoft-developed game is *Ice Thrashers* (working title). It is a futuristic cross between ice hockey, soccer and utter chaos. Set in the "Superconductor Age," players cut along the ice on magnetic skates to rack up points and stay alive. With a shower of ice shards, they'll evade explosive devices that appear in their path and roar up walls to recover the ball. Players travel over ramps and leap bottomless pits, hurdles and jagged ice cracks that conspire to snag unwary blades and condemn competitors to the deep freeze.



ROBOCOP?

Technocop is armed with a computer wrist-watch, criminal radar locator, a hot car and other devices. Back alleys, high-speed car chases and plenty of bad guys make up this game in which five difficulty levels offer plenty of challenge.

Multiple levels take players through rookie challenges and rink riots while superb graphics and joystick controls make Ice Thrashers a unique sports-arcade challenge. The game will sell for $40 and be available by the time you read this.

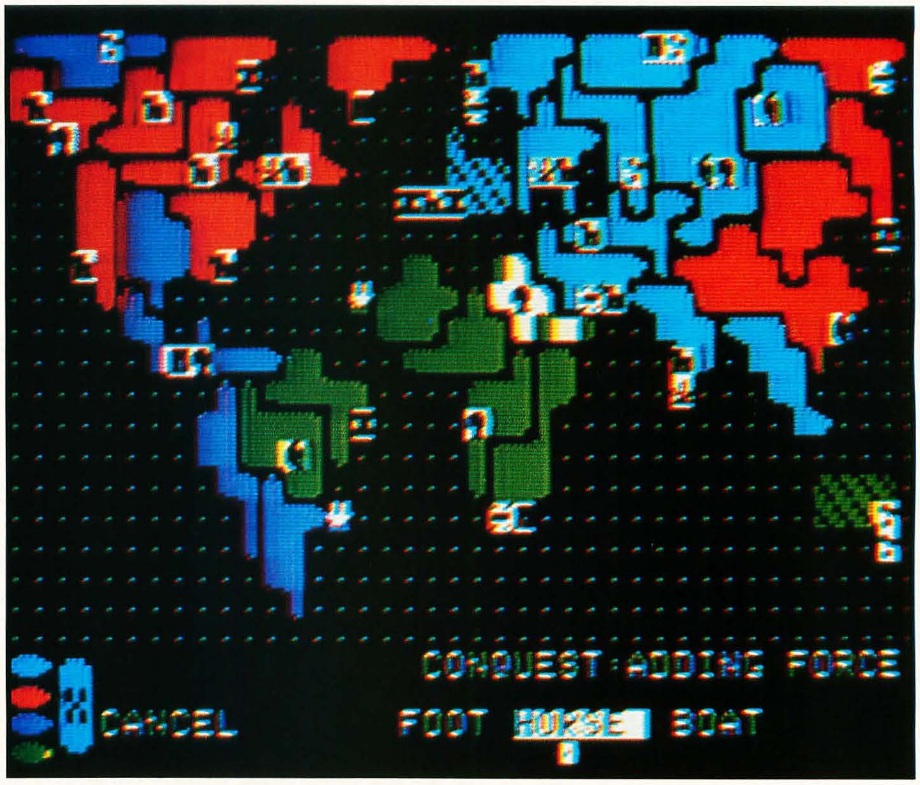
Two of the U.S. Gold games are *Tower Toppler* and *Technocop*. In Tower Toppler, eight dark and deadly towers have risen from the ocean depths on the planet Nebulus. Players must destroy them by setting off destruction mechanisms at the top of each tower. Unfortunately for the player, these towers are well guarded by deadly rolling boulders, flying phantoms, flashing blockades and other obstacles. Players can fire snowball guns to freeze or destroy enemies or ride special elevators to detour the long and hazardous climb, and extra points can be gained by catching fish between the towers. The graphics and 3-D effects in this game are stunning.

The other game is called Technocop. Armed with a computer wristwatch, criminal radar locator, a hot car and other devices, the player's mission is to get the goods on the hoods and earn police badges to upgrade your officer rating. Back alleys, high-speed car chases and plenty of bad guys make up this game in which five difficulty levels offer plenty of challenge.

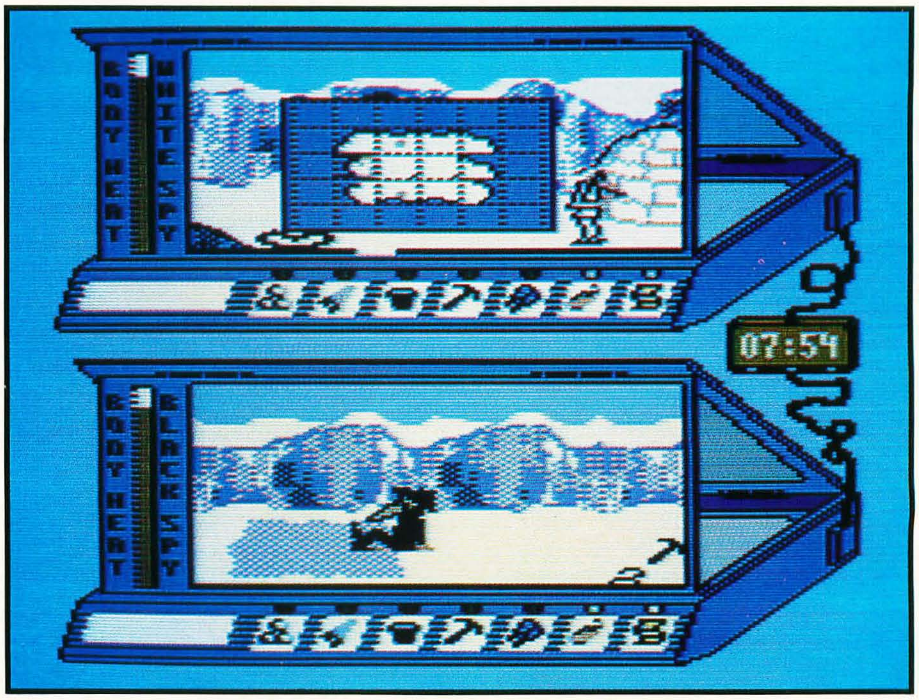
Both Tower Toppler and Technocop will cost $40 and be available this fall.

New Epyx games include *Battleship*, a computerized version of the best-selling board games and *California Games*, in which players compete in such "radical" events as roller-skating, surfing, skateboarding and BMX racing. Awesome, dude. Also new are *Metrocross*, a skateboarding arcade game; *Street Cat*, an urban multi-event game featuring felines; a martial arts game called *Death Sword*; a flight/combat game called *Dive Bomber* in which your mission is to seek out and destroy the *Bismarck*; and *Impossible Mission II*, a newly enhanced sequel to the great original which puts the player in the middle of a high-tech office complex.

Epyx has come a long way from the ear-



LORDS OF CONQUEST



SPY VS. SPY

ly days of the ST in which it was taking a wait-and-see attitude on producing software for the new Atari 68000-based computer. It took its time before deciding to commit to the ST, but now that it has, the wait has been worth it. Epyx deserves your support for its excellent new software titles, and by voting with your checkbook, you can assure that Epyx will continue to release new and even better products.

Accolade, maker of the best golf simulation game, *Mean 18*, was showing *Test Drive*, a driving simulation that allows you to drive five of the hottest cars in the world. The five cars include a Ferrari Testarossa, Lotus Turbo Esprit, Porsche 911 Turbo, Lamborghini Countach and—what else?—a Corvette. A detailed "spec chart" is provided for each of the cars, and authentic graphics, sterling sound and lifelike animation are used to heighten the thrill. Test Drive retails for $40.

The other new Accolade game is *Bubble Ghost*. This clever and imaginative game will fray your nerves as you direct a bubble-blowing ghost through 36 hazard-filled chambers. You'll thrill and chill as the fragile bubble comes dangerously close to walls, burning candles, knives, pins and assortment of other obstacles. Fans located throughout the chambers generate turbulence and even monsters get into the act as they try to sabotage your ghostly efforts. Bubble Ghost was one of the best-looking games at CES and a lot of fun. It is available now and costs $35.

Activision has changed its name to Mediagenic, which better represents its new role as an umbrella company for several software lines, such as Activision, Gamestar, Infocom and Rainbird (formerly Firebird Licensees). The Rainbird label was showing three new titles for the ST. *Starglider II* is a solid 3-D version of the original, great ST game. It is a fast-paced game, combining spectacular graphics, digitized sound effects and smooth animation. It really shows what state-of-the-art ST software can look like.

Players fly around alien deserts, destroying Egron patrol craft and rescuing alien colonies under Egron attack. Multiple planets populated with aliens each possess their own unique characteristics. The futuristic spacecraft is complete with a three-dimensional instrument panel and sophisticated weapons console, and the graphics and animation are smooth and realistic.

Starglider II is the first program released from Rainbird Software that runs on both an Amiga and Atari ST from one disk. It will be available by the time you read this for $45.

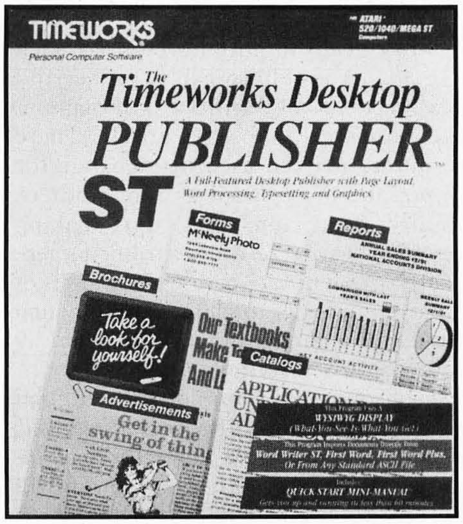
Another new Rainbird game is *Carrier Command*, a strategic warfare simulation that includes arcade action and three-dimensional graphics. The player controls an aircraft carrier with a squadron of remote fighters and an amphibious assault division whose mission is to recapture a fallen carrier, gain control of a strategic island and fight it out with the enemy. You can control up to four amphibious vehicles simultaneously and conduct war maneuvers in a huge territory that includes 64 islands. The game sells for $45 and comes with a fully illustrated mission briefing and operations guide, audio cassette sound track and sticker.

Rainbird also introduced a medieval melodrama-based game called *Black Lamp*. Here, the hero, Jolly Jack, is pitted against a host of animated animals and characters including skull-dropping buzzards, evil eagles and spitting witches. The player's quest is to find the enchanted lamps and ultimately the black lamp which is guarded by a ferocious fire-breathing dragon. Reward for success is the hand of the Princess Grizelda, and the price of failure is death. The price of admission to this once-upon-a-time tale is $25.

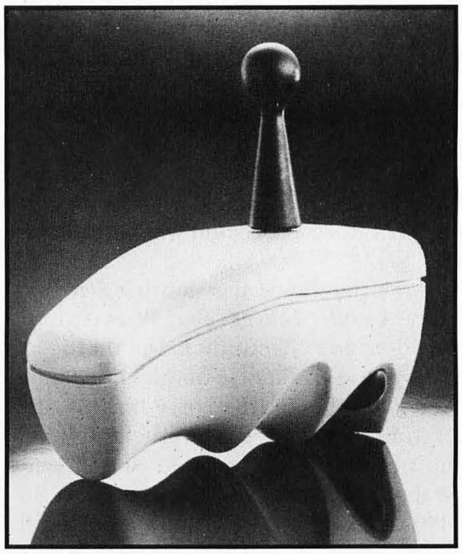
Electronic Arts was showing two new games for the ST. *Rockford* is a sequel to the excellent *Boulderdash* which has been available for the 8-bit Atari computer for a couple of years. In this new ST game, players accompany the world's greatest archaeologist, Rockford, through a series of five exciting locations, each with a unique set of demanding challenges. With the choice of four levels of difficulty, the mission includes the exploration of the Cavern of Craymar, on a quest for the Pharoah's gold pieces. Other locations include the Sunlit Seas of Tiresius where the player must find the fabled Emerald Erasmus and the Kitchens of Kyssandra where you search for the "apples of eternal youth."

In order to escape alive, you (that is, Rockford) must utilize a combination of strategy and speed to collect the required number of treasures in the allotted time. Of course you must avoid the falling rocks, maddening monkeys, fearsome fish and poisonous pizzas. Rockford is now available for $30.

Electronic Arts was also showing *Lords of Conquest*. This game is currently available for $20 and can be compared to the board game, Risk. It is a classic strategy


TIMEWORKS
ATARI 520/1040/MEGA ST Computers
Personal Computer Software
The Timeworks Desktop PUBLISHER ST
A Full-Featured Desktop Publisher with Page Layout, Word Processing, Typesetting and Graphics
Forms
Reports
Brochures
Take a look for yourself!
Our Textbooks Make
Catalogs
Advertisements
Get in the swing of thing
APPLICATION
WYSIWYG DISPLAY
QUICK START MINI-MANUAL

Desktop Publisher ST is a full-featured, fully integrated desktop-publishing program for the ST computer.



The revolutionary design of the Ergostick fulfills the goal of optimizing the operator's hand-eye coordination while minimizing both physical and psychological fatigue.

game in which the action takes place on a world map. The computer adds greater game depth and variety compared to the board game and up to four players can play at once. Each opponent chooses home territories then tries to protect his holdings while conquering territories belonging to other players.

Strategically important resources such as gold, herds of horses, iron, timber and coal are randomly distributed throughout the on-screen map's territories. These resources may be used in exchange for important forces such as foot soldiers, cavalry, fleets and cities, thus making resource-bearing territories prime targets. Lords of Conquest provides four levels of game complexity to choose from, which results in a game's duration of from 20 minutes to several hours.

Electronic Arts is now distributing SSI software. Its new titles include *Shiloh: Grant's Trial, Warship* and *Heroes of the Lance.* All titles are either out now or will be very soon and retail for $40.

Broderbund was showing a fantastic new game called *Typhoon Thompson in Search for the Sea Child.* This is a graphic arcade adventure game written by Dan Gorlin, author of the masterpiece game *Choplifter.* The program's title and story line are strongly reminiscent of an adventure movie, in which our hero, Typhoon Thompson, must rescue an infant boy on a remote planet. In order to accomplish this feat, Typhoon must deal with mischievous sea sprites, ancient technology and spirits from a long-dead civilization.

The 3-D graphics are very realistic and utilize the ST's capabilities well. Further, the mouse control of the jet sled feels just right and adds to the game's verisimilitude. Typhoon Thompson has a suggested retail price of $35 and will be available this fall.

Broderbund was also showing *Star Wars* and *Downhill Challenge.* Star Wars is a fast-paced arcade adventure based on the arcade classic and movie of the same name. You'll have to use the Force for this one. Downhill Challenge is an interactive skiing simulation and a good one at that. Realistic 3-D animation is used, and the player can choose from among downhill, slalom, giant slalom and jumping events. In each event there are separate runs for beginner, intermediate and advanced skiers. Star Wars will sell for $40, and Downhill Challenge will be $30.

A new company, First Row Software, introduced its first ST game called *Prime Time.* This is a humorous role-playing game that gives the players a chance to become TV executives. You run a TV network, cancel shows, do lunch and say things like, "It's all up to you, babe." Prime Time sells for $40 and is available now.

Data East was showing several new ST game titles. *Platoon* is based on the movie of the same name and offers the user with a strategic military combat simulator of the Vietnam experience. As in the movie, and in real life, the game does not have a winner. The user parallels the movie's combat experience in various steps by trying to have his platoon of five men survive the missions, keep their sanity and return to the base intact.

There are six sections to the game, each consisting of increasingly arduous situations. The first section requires the soldier to successfully lead his platoon through a jungle, pick up a box of explosives left by a previous platoon and blow up a bridge. While in the jungle, the platoon encounters armed patrols, booby-trapped trip wires, hidden assassins and deadly snipers. I won't spoil it for you by mentioning the other five sections, but they are increasingly difficult to master.

Other new Data East games include *Speed Buggy,* a racing skills game and *Lock-On,* a flight simulator. Platoon, Speed Buggy and Lock-On are all available immediately and list for $40.

Titus software, a French company, had three new ST games. In *Fire and Forget* you are the commander of Thunder Master, the world's ultimate fighting machine. Your job is to find and eliminate all forces of the Inter-Galactic Liberation Organization. They have guerrillas, mines, bunkers, tanks and helicopters all bent on your destruction. Not to mention their tetranuclear propulsion missiles. You can get help from the magnetic levitation unit, Thunder Cloud, and together you can win the battle to save the earth, all for just $40.

Crazy Cars is a racing game that lets you use a Mercedes 560SEC, Porsche 911 Turbo, Ferrari GTO or Lamborghini Countach. Realistic sound and colorful graphics accompany you as you drive some of the world's fastest cars. There are 72 levels and players start with the Mercedes and work their way up to the GTO. If the players are good enough, that is.

Off Shore Warrior is a driving game of sorts, only the game involves boats instead of cars. It's the future and the extra-terrestrial pacifists have taken control of the economic, social and political life of Earth. Since there is no longer any violence on the planet, the natural aggression of the human inhabitants must be released, and therefore this sport has been created. Armed with one boat and two missiles, all warriors enter the arena knowing that only one can leave alive. All it takes is $40 for you to participate in this futuristic action.

People, Places and Things is an art portfolio with over 240 ready-to-use graphics for all kinds of business, home and school applications.

Mindscape introduced *Captain Blood,* a futuristic science fiction game for the ST ($50). The game uses both fractal and vector graphic techniques to produce some of the most stunning visuals ever seen in a computer game. The program has everything from suggestive humor to complex game play. It seems that Torka, Captain Blood's solar system sweetheart, has promised to bear him 1,000 little Bloods. But there's a hitch. Half a dozen Captain Blood clones are scattered around the galaxy and they're sapping his energy. Your mission is to search the stars for the pesky clones and destroy them. Got it?

In other news, Taito announced that it would be releasing several ports of its arcade games to the ST. These will include *Alcon, Bubble Bobble* and, my favorite, *Arkanoid.* I consider Arkanoid to be sort of a "Breakout Elite." Sierra On-line said that they would be introducing a sequel to *Leisure Suit Larry in the Land of the Lounge Lizards* for the ST called *Looking for Love in All the Wrong Places.*

Spectrum Holobyte will be the first software company to introduce a game created in the Soviet Union to the U.S. market. Called *Tetris,* it should be out by the time you read this. Cinemaware, publishers of *Defender of the Crown,* said it will port all of its games to the ST by the end of the year. These will include *King of Chicago, Sinbad, Rocket Ranger* and *The Three Stooges.*

Some hardware

Camerica has introduced a new remote controlled joystick. Called the *Freedom Stick*, this product features a wireless remote control technology with an infrared system. Players can move up to 20 feet away from a TV screen and can use the built-in automatic rapid-fire switch for high-speed action. Interestingly, the Freedom Stick is the only currently available joystick that is compatible with every major video-game system. Atari, Commodore, Nintendo and Sega games can all be used by this one joystick through the use of the supplied adapters.

The Freedom Stick was developed in Canada by Camerica Corporation and is marketed in the U.S. by Camerica, Ltd. The stick is in the stores now and can be purchased for $70.

Another new joystick was being shown at CES by Wico, the company that pioneered the development and marketing of professional-quality joysticks and controllers in the early 1980s. Called the *Ergostick*, this $25 controller is ergonomically designed to comfortably fit the users hand. The Ergostick is literally formed around the human hand and features a soft, skinlike texture for better gripping. The Ergostick fits easily in both large and small hands and puts the fire button directly adjacent to the index finger. It is meant to be held in the left hand while the right hand operates the vertical part of the stick.

Nongame software

Timeworks was demonstrating its *Desktop Publisher ST* program, a full-featured fully integrated desktop-publishing program for the ST computer. Publisher ST includes fully integrated word processing, page layout, typesetting and graphics functions all in one package. The program appears very easy to use and quite powerful. Some of the major features of the program include a full GEM interface with pulldown menus, icons, scroll bars and dialog boxes, flexible page layout to let you overlap, reposition and resize your text, columns and graphics, built-in fonts that range in sizes from seven to 72 points and high-quality output to a dot matrix or laser printer.

Text can be imported from *First Word Plus, Word Writer ST* or any ASCII file so the program can work with existing word processors. Graphic files can be imported from *DEGAS, NEOChrome* and *Easy Draw* file formats. Further, you can view, edit and layout pages in actual-, double- or half-size windows. Publisher ST is available now and retails for $130. It is clearly the best desktop-publishing program currently available for the ST.

Timeworks also announced that a complete line of desktop-publishing accessory packages will soon be available to work with Publisher ST. These packages include three graphics and illustration packs, two font packages and one package with design and layout ideas for all types of business, school and personal documents. Each accessory package will sell for $40 and is supported by the Timeworks full-time customer technical support team.

People, Places and Things is an art portfolio with over 240 ready-to-use graphics for all kinds of business, home and school applications. *Symbols and Slogans* is another portfolio of 450 camera-ready graphics designed especially for commercial and business applications. *Education Graphics* contains over 290 illustrations, symbols and graphic elements for educational applications from nursery school to university level.

Each of the two Font Packs contain eight unique typefaces, from ten point to 72 point. And with each typeface you can choose bold, italic, underlined, outlined, shadow and super- sub-scripted characters. These additional fonts can be combined with the original Publisher ST fonts to create thousands of type-size combinations from which to choose.

Epyx was showing *Art and Film Director*, the two-in-one package that offers a comprehensive solution to the art needs of computer users. Art Director is a full-featured paint program that uses all of the power of the Atari ST to create dazzling works of art. Menus and icons appear on-screen, so even beginners find it easy to practice at being a master. Computer artists choose from a variety of shapes, lines and colors from the palette, and such features as stretch, bend, bulge, spin and rescale can be used to enhance the pictures.

Film Director then uses true cell animation to transform artwork into stunning graphical presentations. It's easy to automate many of the repetitious steps required by traditional types of animation. A library of music and sound effects is also included in the program to create just the right ambience.

Art and Film Director is a single package that will retail for $80 by the time you read this. The combination of the two programs makes for a powerful package.■

Companies mentioned in this article:

Accolade
550 S. Winchester Blvd.
Suite 200
San Jose, CA 95128
(408) 296-8400

Broderbund Software, Inc.
17 Paul Drive
San Rafael, CA 94903
(415) 492-3200

Camerica, Ltd.
230 Fifth Ave., Suite 1100
New York, NY 10001

Data East USA, Inc.
470 Needles Drive
San Jose, CA 95112
(408) 286-7074

Electronic Arts
1820 Gateway Drive
San Mateo, CA 94404
(415) 571-7171

Epyx, Inc.
600 Galveston Drive
Redwood City, CA 94063
(415) 366-0606

First Row Software
900 East 8th Ave.
Suite 300
King of Prussia, PA 19406
(215) 337-1500

Mindscape
3444 Dundee Rd.
Northbrook, IL 60062
(312) 480-7667

Rainbird
3885 Bohannon Drive
Menlo Park, CA 94025
(415) 322-0412

Timeworks
444 Lake Cook Rd.
Deerfield, IL 60015
(312) 948-9200

Titus Software Corporation
20432 Corisco St.
Chatsworth, CA 91311
(818) 709-3692

Wico Corporation
6400 West Gross Point Rd.
Niles, IL 60648
(312) 647-7500

Software Engineering: MORE TOOLS OF THE TRADE

by Karl E. Wiegers

In our last installment, I tried to persuade you that the right way to begin a software development project is to come up with a precise specification of what the system is to accomplish. In other words, we should study the problem before selecting and implementing a solution. Modern software engineering ideas suggest that a good way to approach a new project is through "structured analysis," a process of building graphical models of the desired software system. The ultimate goal is to create a "structured specification" which we hope will be much more understandable and manageable than a monolithic treatise of text.

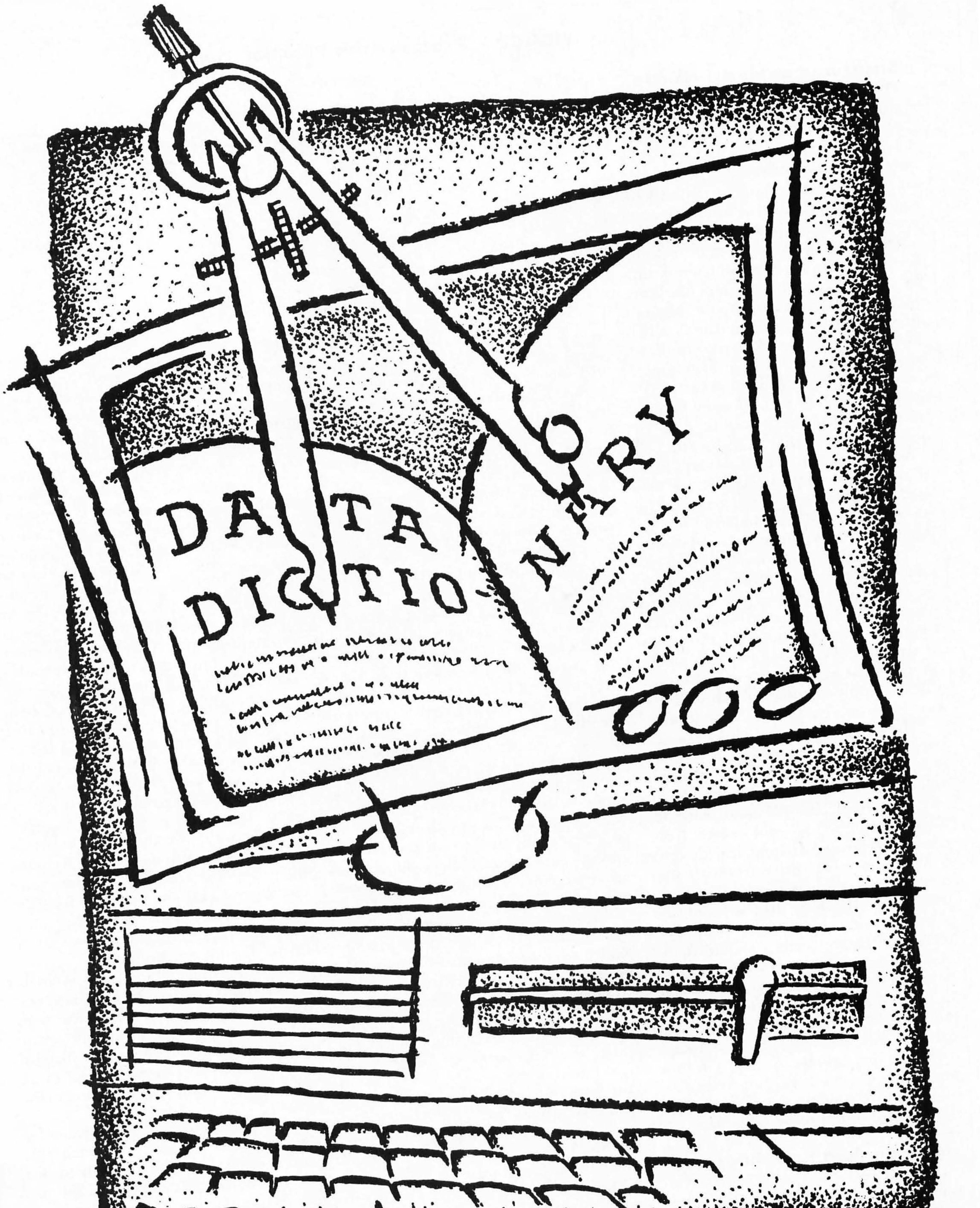
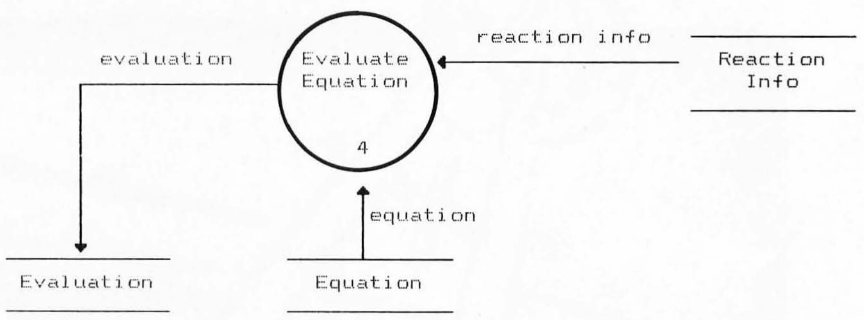


Illustration by John Berado

FIGURE 1. Process 4 for Reaction Time.



Structured analysis: reprise

One of the tools commonly used to construct a system model is a "data flow diagram" or DFD. This is a graphical depiction of the activities carried out by our system (processes), the inputs and outputs of these activities (data flows), organized collections of information (data stores), and objects in the rest of the world which interact with our system in some way (externals). We begin with a very simple, abstract view of our system (the "context diagram"), and decompose that into increasingly more detailed data flow diagrams. This process of "partitioning" or stepwise refinement of our system continues until we've subdivided each high-level process into its individual components. Of course, the definition of an "individual component" is up to you.

Keep in mind that the tools and ideas of structured systems analysis are also used in structured systems design. The two processes are very similar in methodology. During analysis, we are defining all the functions and components of the system we want to build. During design, we are actually defining the individual program modules that will go into our computer program, along with the details of the data connections among them. Most of what we say about analysis tools will also apply to design, which we'll discuss in more detail in the near future.

The idea is to base the design upon the partitioning revealed by structured analysis. While the top-down decomposition will be essentially the same for both analysis and design, the details of each individual component will evolve from conceptual (analysis) to practical (design).

Also, remember that we're discussing methodologies for creating software systems more effectively than we might be able to using older, less structured methods. These "rules" shouldn't be obeyed blindly in every case but should be adapted to situations that are unusual. In fact, there are several schools of thought and sets of conventions for using each of the structured analysis/design tools I'm presenting to you. I'm not preaching dogma here, just trying to share some techniques that many professionals have found to be valuable.

More on data flow diagrams

Last time we looked at the context diagram and zero-level DFD for a real program, a chemistry game I wrote called *Reaction Time.* We covered some of the basic rules of drawing DFDs. A fragment of the zero-level diagram is repeated in Figure 1, showing just Process 4, "Evaluate Equation," and the data flows and stores that connect it with rest of the system. Now you can see why we put data stores between each pair of processes that are connected by a flow of data. By cutting one process out of the diagram along with its connected data flows and stores, we can examine just that one process more carefully, without being distracted by the rest of the diagram.

This is one of the real advantages of using DFDs to decompose a system into small components: We can focus on just one part of the system at a time. While this approach is very useful during analysis, when we're trying to get a handle on the individual functions of our system, it's at least as important during the structured design phase. During design, we're actually defining the individual program modules that will make up our final software system. The modules will be represented by processes on the DFDs, and the data interfaces between modules are represented with the data stores. We'll talk more about this in a future article.

It's very important to give the components of your DFDs meaningful names. Notice from last time that I used names for the processes like "Evaluate Equation," "Change Scores," and "Load Reaction Data." Processes should be given names with a single verb and a single object, not unlike the commands you might supply to a simple computer adventure game. If you find yourself using process names with two verbs, such as "Build and Evaluate Equation," you haven't finished your partitioning task; this process should be split into at least two separate ones.

These may seem like trivial details to worry about. How important can the names of data flows and stores really be? Darned important, if you want to make sure you really understand the structure and function of your system. Vague names like "inputs" for a data flow and "Handle Inputs" for a process aren't meaningful enough to be understood by a non-expert looking at your diagrams. And a vital goal of structured systems analysis is communication. When you've finished this step, your product ("deliverable," in the jargon) includes detailed documentation of all the features and functions of the system. In principle, this collection of words and pictures can be handed to any skilled software engineer to be turned into a computer program.

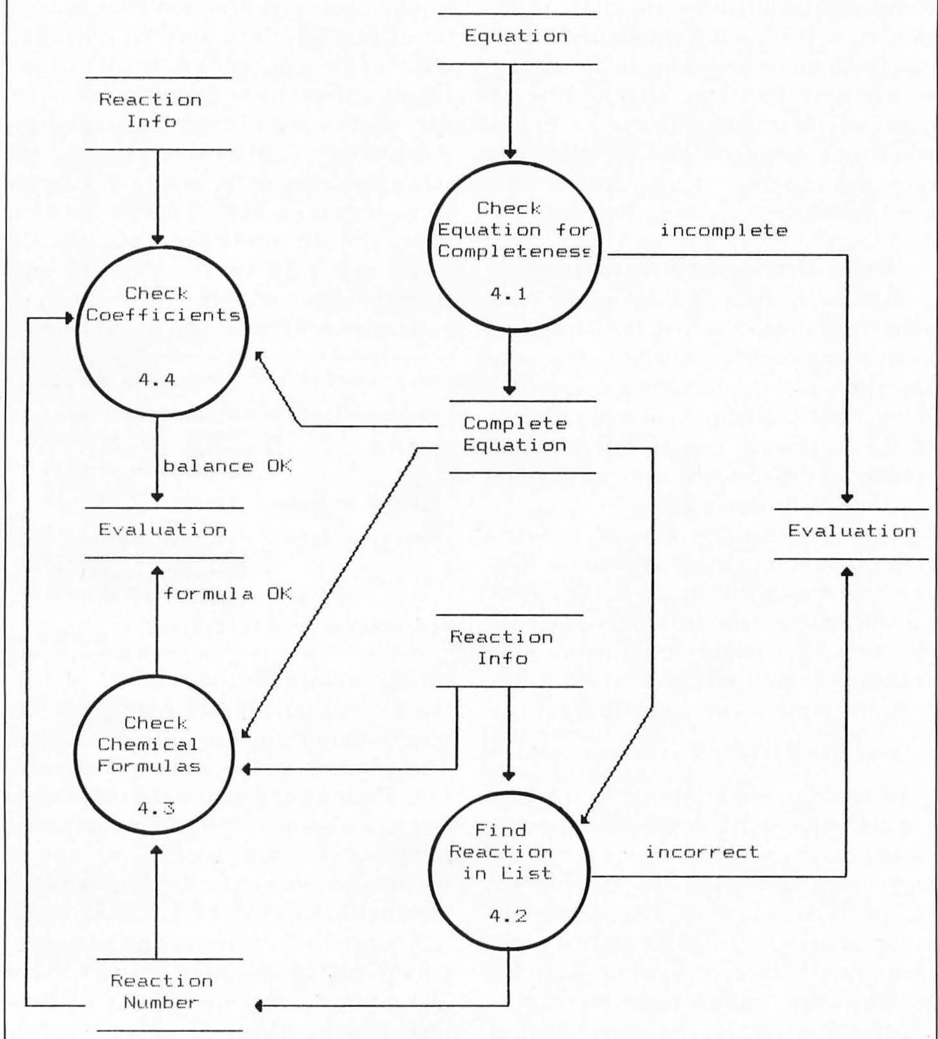
It's also important to make sure that your DFDs are in fact correct. We now need some other Important Rules to keep in mind as we ponder this point (the first four rules were presented in our last article, on system specification and analysis).

Important Rule #5 for DFDs: All data stores should have data flowing out of them. There's no point in placing data into a store if you don't plan to use it again somewhere in the system. Of course, the process that reads data from the store could be on a different diagram from the one that writes to the store. After all, it is through data stores that processes communicate with one another.

Important Rule #6 for DFDs: All data flows should begin and end at data stores, processes, or terminators. You don't want any flows falling off into midair or appearing out of nowhere, because that just introduces an ambiguity (at best) or an error (at worst) into your depiction of the system.

Naturally, following all the rules for drawing DFDs doesn't guarantee that your system is analyzed correctly or that the diagrams make any sense. It just means you followed all the rules, a necessary but not sufficient condition for drawing a useful data flow diagram. Keep the Important Rules in mind at all times!

FIGURE 2. DFD for Expanded Process 4 of Reaction Time.



DFD leveling

As an example of further decomposition, Figure 2 expands the process called "Evaluate Equation" into four daughter (or, if you prefer gender-neutral words, child) processes. These are numbered 4.1, 4.2, 4.3 and 4.4. Children DFDs of process 4.1 would be numbered 4.1.1 4.1.2, and so on. Using such a numbering scheme, you can break any process down into as many levels of detail as you need. In fact, the partitioning process is sometimes called "leveling," and the result is a set of "leveled data flow diagrams." I simplified Figure 2 by leaving some data flows unlabeled; recall our convention that unlabeled data flows take the name of the data store to which they are connected.

Also, I invented a couple of new data stores, "Complete Equation" and "Reaction Number," to represent the communication of information between processes on this diagram level. The stores called "Evaluation" and "Reaction Info" appear in two places, just for convenience of drawing and to avoid having flow lines cross each other. Remember that these data stores are just logical items or collections of data, not necessarily corresponding to physical files.

Important Rule #7 for DFDs: The net inputs and outputs of each child diagram must match those of the parent diagram. All you're really changing when you draw a child diagram is the process, which is now being represented in more detail as several individual processes. Of course, you can create new data stores that are local to the child DFD (or children of that DFD), but the net inputs and outputs must be the same for parent and child. Another way of saying this is that parent and child DFDs must be "balanced" properly.

Notice that, whereas Figure 1 showed the input into the data store called "Evaluation" to be simply a data flow named "evaluation," Figure 2 shows that several different kinds of information can wind up in the Evaluation store. Exactly what sort of data ends up in Evaluation depends on the output from the various processes having flows going into Evaluation. In *Reaction Time* terms, the player's chemical equation can be judged to be incomplete or incorrect, and the flows "balance OK" and "formula OK" both can take on values of either "yes" or "no."

Hmmm. Now we have another problem. We've been drawing more and more diagrams, introducing ever more processes, flows, and stores (but not externals; they all appeared on the context diagram). We are developing a fairly substantial vocabulary of objects in our system. How can we keep them all straight? How can we indicate the individual data elements that make up a composite flow or store? The answer is to use another tool of structured analysis (and design), the data dictionary.

The Data Dictionary

I had a nightmare once. I dreamed I was working on a large software project with two other programmers, using four different computer languages. However, we never told each other what variable names we were using for each item of data in each language, or the exact characteristics of each data item. In this dream, we used four different variable names to represent the user's name in the FORTRAN programs alone! I woke up in a cold sweat at the horror of this situation.

Unfortunately, I was sitting at my desk when I woke up, and the nightmare was the real thing. What was the fundamental problem? Communication. How could we have solved it? With a data dictionary.

A data dictionary is essentially a collection of data about data. (I learned that from the Department of Redundancy Department.) Actually, this makes sense. We need to collect information about the individual pieces of our analysis (or design) into a central repository, so anyone interested in the system can look up the details about every piece it. And isn't this just what a dictionary is?

Our data dictionary for *Reaction Time* isn't too elaborate. We need definitions of each process and external. The data flows and stores themselves will have to be defined, both as to individual data elements as well as how collections of these elements are arranged in each logical data store. Many large systems end up with a gigantic data dictionary. In fact, some companies maintain a corporate data dictionary, so that all the system developers and programmers working in the company agree on the characteristics of each piece of information pertinent to the company's business. This greatly facilitates the creation of new systems that must interface to existing systems.

Data Dictionary notations

To make the data dictionary more readable, some notations have been invented to represent common situations that arise when describing pieces of information. These notations help avoid using phrases in the dictionary entries that just add volume to the text without contributing new information content.

Here's an example. One of the data stores on the zero-level diagram for *Reaction Time* was called "Reaction Files." One way to write a data dictionary entry for Reaction Files might be: Reaction Files consists of seven sets of reaction info.

A much more concise definition is:

```
Reaction Files = 7{reaction info}
```

We've done several things in this simple definition. First, we depicted the definition as an equation, where the equal sign means "consists of" or "is composed of" or "is equivalent to." That simple substitution of one symbol for several words shortens the entry considerably and (I think) makes it much easier to read.

Second, we introduced a new notation, using the curly braces to represent the idea that one piece of data is made up of several repetitions of a smaller piece of data. Using this notation further shortened the definition by removing several more words. This technique continues our philosophy of stepwise refinement or top-down partitioning, by breaking one object down into its component parts. Did you notice that the second definition is much more visually comprehensible than the first? Remember, our emphasis is on effective, efficient communication; all of our software engineering tools are geared toward that goal.

The curly braces can be used to represent several sorts of repetition. Simply enclosing a data item in the curly braces means that an indefinite or arbitrary number of occurrences may be present. The example above indicates that seven (and only seven) repetitions of reaction info make up the Reaction Files store. A notation like:

```
Reaction Files = 3{reaction info}10
```

would mean that Reaction Files is made up of between three and ten (inclusive) reaction infos. So what's a "reaction info"? Figure 3 shows more of our *Reaction Time* data dictionary, further illustrating the partitioning of data and showing the other data dictionary notations that are commonly used. Below I'll write out these definitions in words; see if you like the forms in Figure 3 better than the ones that are pure verbiage.

Reaction info is made up of 16 chemical

```
Reaction Files = 7{reaction info}

reaction info = 16{chemical formulas}
              + number of reactions
              + number of sets
              + 9{reaction data}16

number of sets = [2|4]

reaction data = reaction number
              + 4{compound number}
              + 4{coefficients}
              + (symmetry factor)

coefficients = [1|2|3|4|6]
```

FIGURE 3

cal formulas, plus the number of reactions in the set, plus the number of sets, plus from nine to 16 sets of reaction data. The number of sets can be either two or four. Each set of reaction data consists of a reaction number, plus four compound numbers, plus four coefficients, and an optional symmetry factor. The values of the coefficients can be 1, 2, 3, 4 or 6.

Notice that neither in the preceding paragraph nor in Figure 3 have we actually given descriptions of any of these pieces of information. That's the final part of the data dictionary, and an extremely important part it is. However, I don't need to burden you with detailed descriptions at this point; I think you get the idea.

What we've really done in Figure 3 is describe the "data structures" in our system in a hierarchical fashion. If we continue this decomposition far enough, each item on the right of an equal sign in a definition ultimately will be a fundamental "data element," which must have its own description in the dictionary. Note that *every single data store and data flow* in your system should be included in your dictionary. At this point in the analysis, we don't need to worry about the detailed characteristics (integer, 5-byte character string, etc.) of each piece of data. That sort of implementation detail can wait until the design phase.

Please look again at Figure 3. We already discussed the use of curly braces to depict repetition (sometimes called "iteration"). The data structure for "reaction info" also shows the use of the plus sign to indicate concatenation of two or more components in sequence. Items like "number of reactions," which aren't broken down further in the dictionary, are data elements.

The definition of "number of sets" doesn't show any further decomposition, but it shows yet another operator. The square brackets indicate that the item being defined can take on one of the values inside the brackets that are separated by vertical bars. This vertical bar represents a logical OR operator in many computer systems. In other words, "number of sets" can have one of only two possible values, 2 or 4. Similarly, "coefficients" can take on one of five possible values, as shown in the final entry in Figure 3.

Finally, the definition of "reaction data" shows the inclusion of an optional data element, "symmetry factor," which is enclosed in parentheses. This indicates that one "reaction data" item might contain a symmetry factor, while another may not.

That's it, folks. Only those five notations are necessary for showing a whole spectrum of data dictionary entries. And they can be combined in virtually any way. You could have curly braces around an item that contains a bunch of separate items combined with plus signs. You can have curly braces nested within curly braces. And so on.

Here's how you might use the data dictionary for some project whose details weren't all contained in your brain at the moment. Suppose you looked at Figure 2 and saw the data flow labeled "incomplete" going into data store Evaluation. This tells you immediately that there could be other things in Evaluation besides "incomplete," or else the flow would have the same name as the store. You turn to the data dictionary and find an entry for Evaluation that looks like this:

```
Evaluation = [incomplete | incorrect |
                 balance OK | formula OK]
```

NOTE: Insert appears as one line on your screen.

Now you know that "incomplete" is one of the four possible values that Evaluation could contain. *Voila*; the mystery of the relationship of flow "incomplete" to store Evaluation is revealed.

Realistically speaking, you aren't likely to write up a data dictionary like this for every small project you do. But a data dictionary can be a life-saver if you're working on a large project over a long period of time, or with other programmers, or on several projects with a common focus or data foundation. Plus, it makes great documentation if you have to refer to an old project, of any size, in the far distant future.

How about data dictionary entries for processes? After all, they're certainly objects in our system diagrams. While you should include at least a brief description of each process in your dictionary, the detailed internal structure of a process is defined using yet another modeling tool, called the "process narrative," among other names. We'll look at these more closely in our next installment.

Getting started with DFDs

By now you may well be overwhelmed by the idea of starting your next programming project by drawing a million data flow diagrams and writing a billion data dictionary entries. I've found that a good way to become familiar with these modeling ideas is to apply them to the last program you wrote, before using them for real on the next system. This way you get some practice drawing the diagrams for a system that has already undergone the partitioning process (I hope!) in your head, and at the same time you create some pretty thorough documentation for that older system.

While I have the utmost confidence that you've already documented all your earlier work (right?), some additional pictures can't hurt. And after drawing the appropriate diagrams through this process of "reverse engineering," you should have a better feel of how to use DFDs to depict the essential characteristics of any software system, new or old.

Final words on structured analysis

This pretty much wraps up our introduction to structured analysis, a very important step in modern software development. Remember that the product of analysis is a structured specification, including data flow diagrams and a data dictionary. This specification is really a model (or models, depending on the nature of the project) of the software system you intend to construct. We use a process of stepwise decomposition to identify the essential functions of the system and the movement of data among the system components. This "divide and comprehend" approach makes it much easier to understand a system of any size, as well as facilitating communication with anyone else involved with the project.

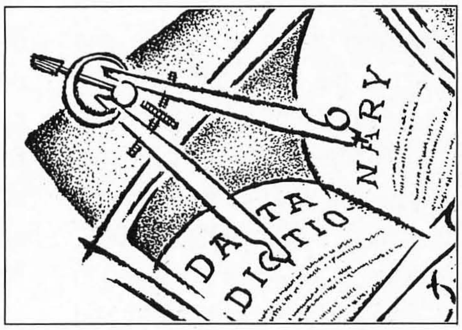
Of course, you don't want to fall into the trap of analysis paralysis, in which you spend all of your time fine-tuning your specification models and never get around to actually building the system. However, pinning down the system requirements helps immeasurably when you get to the point of trying to implement the thing. And it's a lot easier, faster, and cheaper (in terms of your time) to correct errors in the model than it is to change the program code.

From analysis to design

The next step in our software development life cycle is design. If we already have the models from structured analysis, there's no need to redraw them all, because the partitioning is (presumably) okay. You might redo some of the lower-level DFDs to make the processes correspond more closely to specific program modules, as we'll discuss in the near future. Also, the process narratives would be rewritten to be much more implementation oriented.

You might not be dealing with a system so large that a complete structured analysis is required. A small project might be able to get by with an itemized list of the specifications of what the system is intended to do. In that case, you would draw data flow diagrams during the structured design phase. This is very similar to structured analysis at the more abstract levels, but the diagrams take on a physical implementation flavor at the more detailed levels.

In any case, you certainly should draw DFDs before you sit down with a source code editor and start spewing out program statements. The fundamental idea of software engineering is to add a degree of rigor to the process you currently use to develop your software projects. Next time we'll take a look at different ways to describe the internals of processes, and we'll begin to discuss some other aspects of sound, high-quality program design.■



ALICE
The Personal Pascal

An integrated programming environment with 700 HELP screens, an editor that makes errors impossible, and the best GEM interface anywhere. Only $79.95.

"An excellent value." **– Antic**

"It is about as painless a method of learning Pascal as can be devised short of hypnosis. It works!" **– Computer Shopper**

"The product is all anyone could ask for. I would recommend this product to anyone who is considering learning PASCAL . . . or anyone who wishes to prototype small applications which deal closely with GEM." **– ST Informer**

Orders: 1-800-265-2782
Looking Glass Software
124 King St. N. Waterloo, ON. N2J 2X8 519/884-7473

CIRCLE #113 ON READER SERVICE CARD.

LIONHEART
BUSINESS & STATISTICAL SOFTWARE

Explanatory books with professional compiled software; the new standard for statistical use. The influential Seybold Report on Professional Computing has this to say about Lionheart "...our sentimental favorite because of its pragmatic approach to the basic statistical concepts... The thinking is that the computer merely facilitates the calculations; the important thing is to be able to formulate a problem correctly and to determine what type of analysis will be most valuable. Let Lionheart help you get ahead of the competition!

• EXPERIMENTAL STATISTICS	$145
• BUSINESS STATISTICS	145
• MARKETING STATISTICS	145
• QUALITY CONTROL	145
• BIOMETRICS	145
• ECONOMETRICS	145
• FORECASTING AND TIME-SERIES	145
• SALES AND MARKET FORECASTING	145
• PROJECT PLANNING	145
• DECISION ANALYSIS TECHNIQUES	145
• OPTIMIZATION	145
• LINEAR & NON-LINEAR PROGRAMMING	95
• MULTIVARIATE ANALYSIS	125
• REGRESSION	95

SEND FOR FREE BROCHURE

P.O. BOX 379, ALBURG, VT 05440
(514) 933-4918

CIRCLE #114 ON READER SERVICE CARD.

The Writing of Opus

by Doug Harrison

Last month, ST-Log *contained a full-featured spreadsheet program named* Opus. *The article that follows is based on that program and explains some of the programming techniques used in the development of the program. Because of the immense size of Opus's source code, it is impossible for us to offer it on the monthly disk. If you would like a copy of the source code, you may download it from the ST SIG on DELPHI. Those of you without access to DELPHI may obtain a copy by sending your original September or October* ST-Log *disk (make a copy of the disk first; it'll be returned to you with only the Opus source code on it) to:* ST-Log, *P.O. Box 1413-M.O., Manchester, CT 06040-1413. Note: You must include a self-addressed, stamped mailer with your request!*

As you would expect, *Opus* is a very large and somewhat complex program, consisting of 24 source files totaling over 500,000 bytes and 10,000 lines of code. If you should like to dabble about with the program, you will need *Personal Pascal v. 2.01*, a megabyte of memory—and probably a hard drive to make compilation practical. You will also need a resource construction program to alter the resource file and hence the menu and dialogs.

In the course of what follows, I will cite procedures by name. Note also that I have implemented many BIOS, XBIOS, GEMDOS, VDI and AES calls not found in "straight" Personal Pascal; you will need references to these, and I've listed mine at the end of the article. Space restrictions preclude any detailed discussion of the inner workings of Opus, so I'll just consider several of the more interesting routines and techniques that could also have implications for other projects.

General

Like most properly written GEM applications, Opus is event-driven; that is, the program essentially consists of a loop containing a single *Get_Event* call (in C, *evnt_multi*), and specifically, it waits for keypresses, messages and mouse clicks. The main module, OPUS.PAS, contains a loop that calls *window_input*, a general purpose minitext editor containing the *Get_Event* call. *Window_input* echoes typed information to the edit area and exits upon certain keypresses, messages or mouse-clicks, returning *inp_code*. Next, *evaluate_input* is called, and this routine sorts through the various *inp_code* possibilities and calls the appropriate procedure, be it *assign*, *handle_message*, *mouse*, *evaluate_formula* or whatever, and once the designated action is completed, control is returned to *window_input*, and the cycle begins anew.

Window_input also contains the code necessary to control and animate the function menu; briefly, I have used the AES *Wind_Set* call with the value 14, *WF_NewDesk*, to redefine the default object tree for the Desktop to draw as its background. My control panel is part of this object tree, and the menu pulldowns have their *obj_flags* field set to *Hide_Tree*; so they will not be drawn and the AES will not register mouse clicks on the objects.

When the user clicks on the mouse, *Obj_Find* is called to determine if he clicked on one of the menu items. If so, I mask out the relevant pulldown's *Hide_Tree* flag, get the size of the pulldown via *Obj_Size*, blit the background to my 32K blit buffer, and call *Obj_Draw*, the AES routine which draws object trees (which is precisely what a dialog is and what *Do_Dialog* calls). I then enter a loop where *Graf_MKState* (another AES call) tracks the mouse position, and *Obj_Find* returns the object currently under the mouse pointer, which is limited to those in the pulldown.

If *Obj_Find* returns a number other than −1, I know I need to use *Obj_SetState* to set the "Selected" bit in the *obj_state* field of the object, and I may also need to clear this bit in a previously selected object. And when *Graf_MKState* indicates the mouse button is up, the loop terminates, and the chosen item is the one under the mouse pointer at that time. To "clean up," I simply reset the *Hide_Tree* flag and blit the background back. Then it is a simple matter to enter a CASE construct where I check the chosen object and enter its character representation into the edit area.

The nice thing about redefining the Desktop's background is that the Desktop redraws the tree for me whenever it receives a redraw message. (One less thing to worry about!) One problem with this that is not documented in any of my references is the apparent lack of a GEM call to inform the Desktop it needs to use its own, original background when the program terminates. If you don't do this, GEM thinks your object tree is still available (and it's not), but as long as you return to the Desktop, all is well.

However, should you run such a program from a shell, such as the Personal Pascal manager, the Desktop does not get informed of the change when your program ends, and running another program tends to crash the machine as the Desktop attempts to draw a tree whose data structures have been corrupted. The solution? Just before terminating your program, call *Wind_Set* again with *WF_NewDesk*, but pass it a longword zero for an address, rather than an explicit object tree address. This tells Desktop to draw the standard background and keeps everyone happy (my thanks to Charles F. Johnson for this tip).

Data structures

I could have used a simple two-dimensional array of records as a data structure, which would have been exactly analogous to the spreadsheet itself, but this would have severely limited the spreadsheet size and would have made impossible the ability to take advantage of larger than standard memory configurations, as the array dimensions would have been fixed at some small size. In order to provide the large apparent spreadsheet size, I employed a sparse matrix design in which cells containing no data do not "exist" and thus consume no memory.

Essentially, a sparse matrix approach is appropriate when you need an array of very large dimensions that typically is very empty. A sparse matrix emulates such an array through the use of linked lists, a dynamic data structure (one that can grow and shrink) implemented through pointers. Examine Figure 1, which contains the TYPE declarations for the sparse matrix.

The variable *data*, then, is the main variable and it is a 0..999 element array of pointers to records defined by *CellType*, one element per spreadsheet row (row 0 is used as a buffer by various operations, such as MOVE, COPY, SORT, etc). Pointers contain memory addresses and

FIGURE 1

```
TYPE

   DepPtr              = ^DependentCells;
   DependentCells      = PACKED RECORD
                            r,
                            c    : INTEGER; { row and column }
                            next : DepPtr;
                         END;
   CellPtr             = ^CellType;
   CellType            = PACKED RECORD
                              c        : INTEGER; { column }
                              format   : INTEGER;
                              class    : ClassType;
                              status   : StatusType;
                              num      : REAL;
                              str      : ^LorFstr; { label or
                                                     formula }
                              sub      : DepPtr;
                              next     : CellPtr;
                         END;
   DataTable           = ARRAY [0..n_rows] OF CellPtr;
VAR
    data : DataTable;
```

are always four bytes long, so data requires only 4,000 bytes when the program is started. A record is created by calling the standard Pascal procedure *NEW*, as in *NEW(data[1])*, and this allocates space for the pointer type, in this case the record defined by *CellType*.

Counting up the memory required by each field of *CellType* shows that each allocated cell consumes 26 bytes. The new element is referenced as *data[1] ∧ .field*. Now, when Opus detects data entry, it supplies the row and column numbers to *new__cell*, which in turn calls *locate__cell*, which returns the address of the cell if it already exists or NIL. *New__cell* returns the address of the cell, which is easy in the case of the former; if the cell did not exist, it allocates the cell with *NEW* if enough memory is available and then calls *init__cell* or returns NIL. *Init__cell* initializes the fields of the new cell, and it is very important to make sure any pointer fields contain either a valid address of the value NIL.

Note the "next"field in *CellType* is itself of type *CellPtr*; *NEW*ing as in *NEW(data[1] ∧ .next)* allocates a cell whose address is found in *data[1] ∧ .next* and whose elements are referenced as *data[1] ∧ .next ∧ .field*. This can be extrapolated indefinitely, and a linked list is created such that Element 1 points to Element 2 which points to Element 3 and so on.

Since we have a linked list for each row, we store the column numbers for each list member, and we insert cells so that the list is in order by column, we have in effect emulated a two-dimensional array. The end of a list is indicated by a NIL *next*. The only disadvantage is that we can't refer explicitly to the array elements, as in *data[1,1]*; rather, we must perform a list traversal to locate a cell, as follows:

```
FUNCTION LOCATE_CELL ( row,col : INTEGER ) : CellPtr;
   VAR found,passed : BOOLEAN;
       ptr          : CellPtr;
   BEGIN
       ptr := data[row]; { start at beginning of the list }
       found := FALSE;
       passed := FALSE;
       WHILE (ptr <> NIL) AND (NOT found) AND (NOT passed) DO
          IF ptr^.c = col THEN
             found := TRUE { yea! }
          ELSE IF ptr^.c > col THEN { if we passed it up, we never }
             passed := TRUE         { will find it, since we store }
          ELSE                      { elements in order }
             ptr := ptr^.next; { on to next element }
       IF found THEN
          locate_cell := ptr
       ELSE
          locate_cell := NIL
   END; { LOCATE_CELL }
```

Note that *found* is TRUE if and only if the cell exists. Of course, you must also write procedures to delete elements from the list, and you call the standard procedure *DISPOSE* to return the memory to the system, so that it may be reused.

Now, take a look at the field called *Sub* in *CellType*, which stands for subordinate. *Sub* is itself the first element in a linked list of dependent cells—that is, cells holding formulas that reference the cell. After entering a formula and evaluating it, *all__lists* is invoked to update all pertinent dependency lists, and it begins by calling *scan__for__cells*, which returns the position(s) of cell references within the formula. *All__lists* then calls *translate__cell*, which converts the cell reference to row and column numbers, and then it supplies *list__insert* with these values as well as the row and column number of the cell containing the formula.

A *common problem in programs such as Opus is the need to evaluate or parse mathematical expressions.*

List__insert then calls *new__cell*, supplying the row and column for the referenced cell, and that procedure creates the cell and returns its address or returns NIL, as explained previously. Now, *list__insert* (provided *new__cell* did not return NIL) either adds the row and column of the cell containing the formula to the referenced cell's dependent cell list or creates the list. Each entry requires eight bytes, as can be seen by summing the data sizes for the fields in *DependentCells*.

Now, after changing the cell's value, it is simple to traverse the dependent cell list and recalculate just those cells that are affected by the change, as Opus knows these cells by their row and column numbers. These cells may themselves have dependent cell lists, which also will be traversed. Hence, a change to a single cell may cause recalculation of any number of cells, but just those cells that need to be recalculated. Again, specific procedures are required to locate, create and delete list members and also find the end of the list.

The cell format is stored in the *CellType* field format, which is a word where the status of various bits indicate the various format options. For example, to test whether the cell is to be displayed in bold type, I say:

```
IF ptr^.format & bold_mask <> 0 THEN
   display in bold type.
```

The values for the various masks are found in GCTV.INC (Global Constants, Types and Variables), and some pertinent routines that illustrate their use are *change__format*, *find__prec*, *find__just* and *draw__cell*.

Now, since we know the amount of memory required by cells in various states and by dependent cell list entries, it is rather simple to keep track of available memory, so that we avoid the dreaded crash. Thus, Opus checks the global variable *working__memory* before performing a *NEW* action. After performing a *DISPOSE*, the amount of memory released is added back to *working__memory*. Free memory in Pascal is organized into the stack and the heap, and these two entities grow towards one another as local variables are allocated from the stack and dynamic or pointer variables are allocated from the heap. If these collide, you have a "stack overruns heap" crash, and so the importance of keeping track of memory allocation is obvious in a dynamic environment.

The function *MemAvail* is used to get the initial amount of free heap/stack space, and its use is only valid before allocation of any variables other than the global ones; I've experienced this, and OSS has verified it. My *working__memory* then is computed as follows:

```
original_memory := MemAvail*2; { *2 to get bytes }
working_memory := original_memory-20000;
```

In effect, this reserves 20,000 bytes for the stack, as Opus will never allocate memory such that *working_memory* becomes negative. This is important, because some procedures need several Kilobytes for local variables and also because the expression evaluator uses recursion, which may mean "stacking" many local variables. Twenty kilobytes should be adequate almost all the time, but under some circumstances it isn't.

For example, consider a spreadsheet where cell A1 has a dependent A2 which has a dependent A3 and so on for 1,000 cells. If automatic recalculation is in effect, *evaluate_formula* will call itself 1,000 times before it starts "returning" and thus deallocating space for all the local variables, 20 to 30 bytes per call. The recalculation in and of itself requires 20 to 30K just to operate, and if this much memory is not available, the ol' F1 key should be relabeled "Stack overuns heap—CRASH!" because that is precisely the effect pressing it will have.

It would be possible (but extremely tedious) to check memory before doing a recursive call, much like I do before allocating cells. However, it would be hard to say where to draw the line at checking procedure calls, since many are recursive, but some are not, and since this would so rarely be a problem as you would basically have to create a very contrived worksheet to provoke this kind of crash. Thus, I don't do it. However, if you really need to create a worksheet with the characteristics described above, or you simply are not sure, you can always turn off both automatic recalculation and natural order and preclude the possibility of this type of problem.

In summary, the main data structure is an array of linked lists, the members of which are cells of record type *CellPtr*, and each cell may contain a linked list for dependent cells. Sounds complicated, and it is, especially when it comes to deleting cells. For more information on pointers and linked lists, see the bibliography at the end of this article.

The expression evaluator

A common problem in programs such as Opus is the need to evaluate or parse mathematical expressions. We typically write these expressions in a notation known as infix; that is, the operators (+,–, etc) occur between the operands (numbers, etc). The computer, however, prefers postfix notation, where the operators follow the operands (1+2 infix = 1 2 + postfix). The evaluator must also provide for operator precedence, as discussed previously, so that 1+2*3 = 7 and not 9. Several parsing techniques have been devised, and I originally used one similar to that used by 8-bit Atari BASIC, described in the COMPUTE! book, *The Atari BASIC Source Book*.

However, over the course of several months, I became more and more dissatisfied with my parser, as the syntax checking and error recovery were less than ideal. Plus, it was quite difficult to understand, so much so that if I went more than a few days without looking at it, I would completely forget how it worked. So, I began searching through my old computer magazines and encountered an article in *Computer Language*, called "Parsing an Equation without Recursion" (June 1987). The author described a method quite similar to mine, and it was just as hard to understand. Fortunately, he had done some prior research and cited one reference to an article in *BYTE*.

A trip to the local library proved most profitable, as I discovered the method I would ultimately use in an article in the August 1985, *BYTE*, written by Jonathan Amsterdam and entitled "Context-Free Parsing of Arithmetic Expressions." Browsing through subsequent issues led me to, of all things, an article by the same author entitled "Build a Spreadsheet Program" (July 1986, *BYTE*), which was of further assistance, especially since he provided Modula 2 source code for a very minimal parser, which I found on BIX, the *BYTE* information network.

In these articles, Mr. Amsterdam described a recursive-descent parser based on a formal grammar, stated in Backus-Naur notation. This technique is also used by many compilers. You may also recall the syntax definitions in the original Personal Pascal manual were presented in this Backus-Naur formalism. I was immediately impressed with the simple elegance of the method and began work adapting it for my program.

Examine Figure 2, which contains my grammar definitions. As an example, the first line is read as: "Full expression is defined as being either a value expression or a boolean expression." Defining the grammar in this fashion both provides the desired operator precedence and suggests the algorithm. We create procedures entitled *full_expr*, *val_expr*, *term* and *factor*, all local to the main procedure *evaluate_formula*, and these procedures are more or less direct implementations of the grammar.

FIGURE 2

```
A. The terms on the left have cooresponding procedures

   <FullExpr> ::= <valexpr> | <boolexpr>
   <ValExpr> ::= <term> { <addop> <term> }
   <Term> ::= <factor> { <mulop> <factor> }
   <Factor> ::= real | <cellref> | <function call> |
                ( <fullexpr> ) | -<factor>  { ^<factor> }
   <BoolExpr> ::= <valexpr> <boolop> <valexpr>

B. Definitions

   <CellRange> ::= <cellref:cellref>
   <CellRef> ::= {$}A{$}1..{$}IU{$}999
   <AddOp> ::= + | -
   <MulOp> ::= * | /
   <BoolOp> ::=  >  |  <  |  >=  |  <=  |  <>  |  =

...plus the various functions with their argument
requirements already stated.
::= means "is defined as."
| means "or."
{ } means "optional, and may be repeated".
```

Some of the important variables are *str*, the string to be parsed, *str__pos*, the position we are currently at within the string, and *len*, the length of the string. Obviously, we can parse as long as *str__pos* < = *len*, and I check for that before any attempt to access a character, typically as "str[str__pos]."

Now, *evaluate__formula* begins by calling *full__expr*, and *full__expr* immediately calls *val__expr*. *Val__expr* calls *term*, and *term* calls *factor*. *Factor* then decides whether the substring beginning at *str__pos* fits the pattern for a real number, a cell reference, etc., and calls the appropriate routine to convert a string to a real, decode a cell reference, and so on. If a potential function name is encountered, it calls a routine to check the function list (via binary search) to see if there is a match.

If an open parentheses is found, it calls *full__expr*, and immediately upon return checks for the required close parentheses. If *factor* successfully terminates, the variable *result* will hold the real-valued result of the evaluation. Then, as for any subprogram, control returns to the routine that called it, namely *Term*, and *str__pos* now indicates the position immediately following the factor. *term* then enters a loop where it checks for a MulOp (remember a Term is defined as a factor followed optionally by one or several "MulOp factor" pairs).

If it finds one, it stores the type of MulOp encountered and again calls *factor*, and when *factor* is done, *term* either multiplies or divides the two results. The loop keeps repeating until no MulOp is found or an error condition occurs. Control returns to *val__expr*, which enters a similar loop, except that it checks for an AddOp. When *val__expr* terminates, *full__expr* checks for the presence of a BoolOp. If one is found, it stores *result* in a local variable and again calls *full__expr* to evaluate the right side of the expression. This ensures the boolean operators have the lowest precedence, and as you may have noted, correct precedence for the AddOps and MulOps has also been established.

In a nutshell, this is how the expression evaluator works. Rather than provide a step by step walk-through of an evaluation, I ask that you consult the references I cited or better yet, examine my source code. Note that these routines are all mutually recursive, hence the FORWARD declarations.

Lagniappe

Opus contains far more interesting considerations and routines than I have space (or time!) to describe. I have extensively commented some you may find of general use, including *my__exp* and *my__ln*, which are accurate replacement functions for the library procedures that ensure calculation of the financial functions accurate between nine to ten digits, which would otherwise be accurate between only five to six digits. These use a combination of techniques, including one for fast calculation of integer powers found in "Computer Approximations" in the April 1986, *BYTE*, and my own implementation of power series approximations optimized by taking advantage of some logarithmic and exponential properties.

Other useful language extensions include my routines for real number and string conversions, aptly named *real__to__string* and *string__to__real*. *Real__to__string* is very flexible and allows you to specify the number of decimal places and whether scientific notation is to be used.

The Writing of Opus

If you're using a resource construction program, which I highly recommend, don't use *Set__DText Set__DEdit* and *Get__DEdit*. Rather, avoid redundancy, add elegance, and save a few kilobytes by using *Set__Text* and *Get__Text*. Use *Map__Tree* to deselect all selectable items within a dialog with a single call, rather than calling *Obj__SetState* for each one. Also use *Map__Tree* to return the one selected item from a group of selectable items, rather than explicit "IF Obj__State(tree,item) & Selected < > 0 THEN. . ." constructs for every possibility. My acknowlegements to Tim Oren and his "ProGem" series for these latter routines.

I designed Opus to be expandable through the use of desk accessories (DAs). Essentially, a DA may communicate with Opus, requesting certain information, and Opus will respond by writing a message containing the desired information to the DA. In the future, I may use this feature to add graphing, which won't require a new version of Opus itself.

Epilogue

Thus concludes our tour of Opus. I hope I anticipated most of your questions, and I further hope you find Opus a truly useful, enjoyable tool. Should you have any questions, comments, or criticisms, I may be reached at:

DELPHI: MEDSTUD
CompuServe: 72277,2315
GEnie: DougH.

I would also be most pleased to hear about your application of my program, and I would enjoy seeing any of your templates, so upload them for all to use.

Bibliography:

Aho, Alfred V., John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms.* Reading, Mass.: Addison-Wesley, 1985.

Amsterdam, Jonathan. "Context-Free Parsing of Arithmetic Expressions." *BYTE*, August 1985, pp. 139-144.

Amsterdam, Jonathan. "Build a Spreadsheet Program." *BYTE*, July 1986, pp. 97-108.

Magruder, Scott. "Parsing an Equation without Recursion." *Computer Language*, June 1987, pp. 45-48.

Moshier, Stephen L. "Computer Approximations." *BYTE*, April 1986, pp. 161-178.

Wilkinson, Bill, Kathleen O'Brien, and Paul Laughton. *The Atari BASIC Source Book.* Greensborough, N.C.: Compute! Books, 1983.

Atari ST references used:

Oren, Tim. *ProGem.* Available on DELPHI, CompuServe and GEnie.

Leemon, Sheldon. *Atari ST: VDI.* Greensboro, NC: Compute! Publications, Inc., 1987.

Leemon, Sheldon. *Atari ST: AES.* Greensboro, NC: Compute! Publications, Inc., 1987.

Gerits, K., L. Englisch and R. Bruckman. *Atari ST Internals.* Grand Rapids, MI: Abacus Software, Inc., 1985.

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

SAVE TIME AND MONEY SUBSCRIBE TO ST-LOG

SAVE $14 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ST-LOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS! GET THE MOST OUT OF YOUR COMPUTER

SUBSCRIBE TO ST-LOG TODAY!

☐ 1 YEAR @ $28 — SAVE $14! (MCJWW)
FOREIGN — ADD $7 PER YEAR
☐ 1 YEAR *WITH* DISK @ $79 (DCJWW)
FOREIGN — ADD $15 PER YEAR
☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MC #____________

EXPIRATION DATE SIGNATURE

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

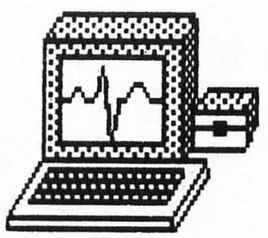
NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires December 30, 1988. Your first issue will arrive in 6 TO 8 weeks.

WATCH FOR IT!



An incredible simulation

Cardiac Arrest!

with binder and manual, $69.
See discounted package price.

Cardiac Arrest! is a unique product. In this mathematically-based simulator, you interpret the history, on-screen EKG, lab data, and vital signs, then give treatment orders in plain English. While many computer users enjoy Cardiac Arrest! as a challenging medical adventure game, it's a sophisticated product used world-wide for ACLS (Advanced Cardiac Life Support) education. **IBM, Apple II+/c/e, Atari ST, Atari XL/E.**

Antic: "impressive and amazingly complete"
ST World: "both highly educational and fun to play"

We support our products. Updates will be available to users for $6 each when ACLS recommendations change. Our software is NOT copy-protected.

Cardiac Arrest!	$69
ACLS Protocols	$29
EKG Teaching	$29
CardioQuiz	$19
Blood Gases	$24
QuizPlus	$29
Demo	$7

Ask about the four-disk ACLS Package (includes Cardiac Arrest!) for $109. Order direct!

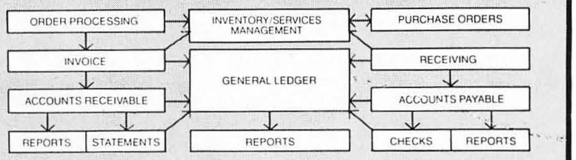
Mad Scientist Software
2063 N. 820 W., Pleasant Grove, UT 84062
Visa/MC orders call 801-785-3028

CIRCLE #121 ON READER SERVICE CARD.

Trade the Rest for the B.E.S.T.!

One program powerful enough to handle all of your basic business needs! Designed for people who are NOT accountants or computer experts.

"A TRULY INTEGRATED ACCOUNTING SYSTEM"



— LIMITED TIME OFFER —

Trade in ANY business accounting program for $50 credit on B.E.S.T. Business Management!

B.E.S.T. Business Management is available for the Commodore Amiga, Atari ST, and MS-DOS.

Offer good only at participating dealers!
Offer expires October 31, 1988

— MONEY BACK GUARANTEE —

Call for more information!
Call 1-800-289-2544 for the dealer nearest you!

CIRCLE #122 ON READER SERVICE CARD.

Spectral SORCERY

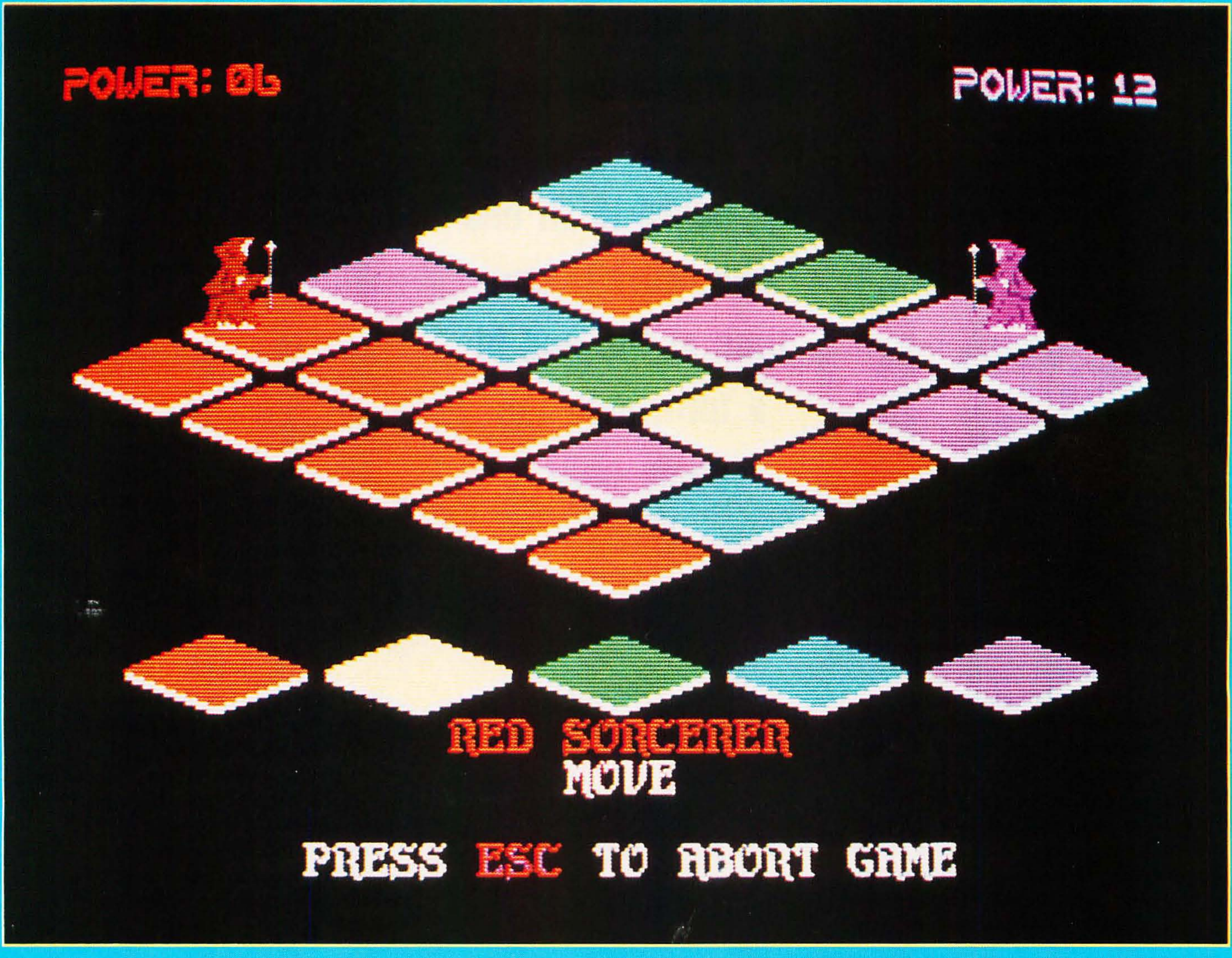
by Jeff Makaiwi

Spectral Sorcery, as the name might suggest, is a game based on colors and a form of magic. It is a strategy board game for zero, one or two players, the game being played on a partial-perspective playing board that consists of a 5x5 matrix of colored squares (pads). The color of each pad can be changed by the players to any one of the five colors in the game's spectrum (red, yellow, green, blue and violet). The object of the game is to convert all of the pads on the board to your color (red or violet, depending on which side you are playing). To accomplish this task, you are provided with a single playing piece (your wizard) and a number of actions that the wizard is capable of performing.

The best way to learn to play this game is to watch it in action. So, boot it up and mess around with it. There is online help for just about everything, so you probably won't need to read the rules.

The best way to learn to play this game is to watch it in action. So, boot it up and mess around with it.

Note that, because of the large size of this program, the listings could not be included in the magazine. The complete program is available on this month's disk version or in the ST SIG on DELPHI.



Boot-up

The game starts with a title/credit screen. If you don't press the space bar (as it suggests), the game will switch to the Options screen after about one minute. From the Options screen, you can set the game controllers and several start-up options (including the board configuration). If you have the mouse plugged in, make sure you don't leave either of the players under the control of Joystick 0. If you do, strange things may occur since the mouse rollers do send signals in along the same lines as the joystick—the wrong ones.

After about a minute of complete keyboard idling, the game will automatically start a demo using all of the settings just as they were set before the demo started. You might want to watch a demo before playing, just to get an idea of how the game works. You can interrupt the game with the escape key at any time. Also, the messages that are displayed are intentionally slowed down; you may press the space bar at any time to make the program skip past a message display pause.

Controls

This game is controlled mostly via Joystick. However, all Options are set using the keyboard. To start with, you set all of the game options with the function keys (F1 to F6), you start the game with the space bar, you get a brief instruction screen with the help key and you use the escape key to abort a game or exit the program entirely. Also, during play you can press the help key to get a description of the action or spell that you want.

The joysticks are used to directly control the wizard's movements and to select the actions in the text window. To select an action or a spell, you move the stick forward or backward (to scroll the list forward or backward). The name of the action or spell will appear on the second line of the text window (at the bottom of the screen), just below the player identification line (red/violet). To pick one, you just press the stick button. For information on how to use an action or spell, press the help key (before you actually select the spell with the stick button).

When using the joysticks for movement, Up moves the wizard (or teleport target pad) up and to the left; Down moves down and to the right; Right moves up and to the right; and Left moves down and to the left. Try it, it will become easy.

The object of the game is to convert all of the pads to your color and to hold that configuration until the end of a turn. If the board is all one color at any one time at the end of a turn, then the game ends.

Overview

This is a game of strategy. The field of play is a 5x5 board of squares. Each square consists of a colored surface (the pad) and a colored edge (the edge). Each pad may, at any given time, be any one of the five spectral colors. (Okay, so there are seven colors in the standard spectrum. Cut me some proverbial slack; I only wanted five.) The order of the spectrum colors is shown in a row of platforms just above the text window. From left to right they are: red, yellow, green, blue and violet.

The edges come in three colors, where each color indicates the "lock" status of the square. A gray edge indicates that the square is not locked (its pad color may be changed). The other colors (InfraRed and UltraViolet) mean that one of the two players has locked the pad color so that its color cannot be changed until the lock is removed. (More on this in the spells list.)

The object of the game is to convert all of the pads to your color and to hold that configuration until the end of a turn. If the board is all one color at any one time at the end of a turn, then the game ends. If the board is all red or all violet, then the appropriate player wins. If it is one of the other colors (yellow, green or blue), the game ends in a draw.

This last circumstance rarely occurs since all of the options available to the player are designed to convert things to his or her color.

The game consists of an indefinite number of turns. Each turn is broken down into a series of rounds. In each round, both of the players are allowed to take one (and sometimes two actions). After an action is selected, the appropriate amount of power is deducted from the player's allotment for the turn. When both players are out of power (or both choose to pass), the turn ends, some bookkeeping is performed, and new power is allocated based on the configuration of the board at the end of the turn. Essentially, a player gets 16 base-power points plus one point for every pad of his/her color on the board at the end of the previous turn. One point is then deducted for every lock belonging to the player that was on the board at the end of the previous turn. Minimum power is always one point; the maximum power is 99 points (assuming that you use the Save Power option).

Actions

Now that you have an idea of what you are supposed to accomplish, you may be wondering how you go about it. Well, if you have ever played *Archon* or *Archon II* from Electronic Arts, you will recognize the overall approach I have taken. At the bottom of the screen, there is a four-line text window where you will see the title of the player who is currently moving, the action that is currently selectable and a few other messages and pieces of information. You might want to scroll through the text as you read the descriptions below. Also, remember that you can get on-line help as well.

Move (default): This action allows you to move your wizard from the platform it is currently on to any one of the eight adjacent platforms. After you select Move, the joystick will allow you to select the pad to which to move. If you try to move too far from your starting pad, or if you try to leave the board, a warning will be printed in the text window, and the attempt will not be executed. When your wizard is positioned on the desired destination platform, press the joystick button and the move will be finalized.

While you are moving, the color of the platform you are on will be printed in the text window. If you return to your starting platform, the word "CANCEL" will be printed in place of the color. If you press the button while "CANCEL" is displayed, your move will be canceled, and you will be allowed to select another action.

The power used for the actual move is one point. Additionally, if you have more than one point of power when you start your move, and you are moving to a platform that is not your color, an additional point will be expended and the color of the platform will be "tweeked" one level closer to your own. If the platform is already your color, or you only had one power point, then no color change takes place (and the extra power point is not expended).

Cast: This action activates the sub-list of spells. You then select one of the spells (in the same way you selected the Cast action). If you don't want to cast a spell, then you select the NO SPELL item. See the Spells section below for more info on what they do.

Stomp: This action causes the wizard to bounce on the platform he currently occupies. If the pad is not your color, then it is tweeked at a cost of one power point. If the pad is your color, no power is used, and the pad color does not change. It is essentially a null action.

What you are playing against, in a very real sense, is the board itself. If you are a chess master, it might be a piece of cake to tromp.

Help Me: This action directs the computer to select your next move for you. It uses the same routines that select actions for a computer-controlled player. After the action is selected, control returns to the player. Once this is selected, you can't cancel the action that the computer selects for you.

Resign: A graceful way to get out of the game. If you feel that you cannot win, just select this, and the game will end as if it was a tie. However, the fact that you resigned will be displayed (rather than the announcement of a draw).

Pass: Take no action for this round. It does not end the turn unless both players select to pass within the same round. Also, if one player is out of power at the start of a round (regardless of who moves first), and the other player selects Pass, the turn will end. However, if the first player passes and the second player takes an action that uses all of his/her power, the turn will not end in that round.

Spells

Spells can have a drastic effect on the configuration of the board (color layout, that is). Just keep in mind that the spells are all centered around your wizard. So, any reference to the origin of a spell refers to the location of the wizard at the time that the spell is cast.

Convert: This changes the color a pad (through a series of Tweeks) to the player's color. It costs one point per each Tweek that is required to reach the player's color. So, the cost can range from one to four points. If you cast this spell on a pad that is already your color, the pad color wraps to the color of your opponent (at a cost of one point). This has its uses, especially when you want to setup for a "Blitz."

Teleport: This spell lets you move your wizard to any other platform on the board (if you have sufficient power). The cost is based on the color of the destination platform. Teleporting to a platform of your own color costs one point, while going to a platform of your opponent's color costs five points. Figuring the cost of teleporting to the intermediate platform colors is left as an exercise to the reader.

When you activate this spell, a multicolor pulsating cursor will cover the surface of the origin platform. You move it around just as you would move a wizard, but you have no range restriction. As you move it, the color of the pad that is covered will be displayed. If you return the cursor to the origin platform, "CANCEL" will be displayed. When you press the button, the computer will check to see if you have enough power to teleport to the desired destination (or cancel the teleport if you picked that). If you don't have enough power, you get a message and have to pick a different spot. If your power is sufficient, you get whooshed off to the new location.

Blitz: This spell is kinda hard to describe. (I'm a linguist, so I can use words like "kinda.") Officially, it is a recursive Tweek that expands laterally and vertically from the origin. As it expands, it treats each of the platforms that it affects as if it were also an origin platform. In words, it looks at the color of the origin platform and remembers it, then it tweeks the origin platform. Next, it looks at the four platforms that are adjacent to the origin (diagonals are not adjacent for this). If an adjacent pad is of the same color as the origin was before it was tweeked, its location is placed in a queue. After all adjacent platforms have been checked, the first platform entry in the queue is retrieved and that platform becomes the new origin. The procedure above repeats until the queue is empty.

When the queue is emptied, the program returns to the original origin and attempts to repeat the entire process with the new set of colors. The end conditions are: (1) all power used (since each tweek uses one point), (2) the origin and all platforms in the queue are already the player's color. One implication of (2) is that casting Blitz on a platform of your own color will have absolutely no effect. An implication of (1) and (2) is that this spell can quite easily (and often does) use all of your power in one massive shot. Also, since this is such a powerful spell, I have made the assumption that the wizard must have a certain amount of energy to "ignite" the spell. You don't necessarily use all of the power, but it has to be there in potential, in order for the spell to get its initial kickoff.

If this description is not clear, I suggest you put the game into two-player mode and experiment with Blitzing different places. You will get the idea after a while.

Transmute: This is a straightforward spell that affects the large group of nine platforms that are centered around the wizard's platform. If you're on the edge of the board, then it may only affect four or six platforms. What happens is that each platform in the transmute zone is tweeked in the same way they would be tweeked if you were to stomp on each one of them. The cost is one point per platform that actually undergoes a color change.

Also, since this too is a rather far-reaching spell, it requires the player to have a nine-point power reserve in order to "ignite" it. Once again, this ignition power is not actually expended, it just has to be there at the time the spell is engaged.

Lock: This places a lock on the platform occupied by the wizard. A locked platform's color cannot be changed by any means until the lock is removed. The cost depends on the color of the platform relative to the color of the player. Platforms far from your color cost more to lock than platforms close to or of your color.

This is a good way to block a Blitz, since a locked platform functions as if it is a totally unique color (that does not match any other color on the board and therefore cannot be Blitzed). Also, if Convert or Blitz are cast while the wizard is on a locked platform (regardless of who owns the Lock), the spells will fail, and the player will lose his/her action for that round.

At then end of a turn (not just a round), all locks are counted. For every lock remaining at the end of a turn, the player will lose one point of power from the pool of points that are awarded at the start of each new turn. Since it is theoretically possible for a player to have more locks than power points (16 base points – 25 locks), there is a provision that a player always be granted at least one point of power (to cast the Release spell).

Unlock: This removes a Lock from the platform occupied by the wizard. If the Lock is one of your own, there is no cost. If it is a Lock placed by your opponent, the cost is just as if you were placing a Lock. However, in addition to that standard power requirement, there is an adjustment made based on who has advantage on the board. For every five platforms that the player has over and above the number of platforms that his opponent has, one point is added to the cost of the Unlock spell. What this means is that it costs more for the player with the board advantage to remove Locks than it does for the disadvantaged player.

One special feature of this spell is that it does not count as an action. If you Unlock a pad, you can still take another action or cast another spell.

Release Locks: This spell removes all of your Locks from the board at a cost of one point. It is here in the off chance that you somehow manage to get so many locks on the board that you don't have any power to do anything. Since this has yet to happen (in over two years of playing this game), I don't know if this is really a useful spell.

No Spell: This is what you pick if you don't want to cast a spell. It returns you to the Action selection list, doesn't cost anything and you don't lose your turn.

About your 68000-based opponent

The routines I wrote to make the computer play against you are extremely simple. Nevertheless, you will probably lose the first few times you play against it. Of course, if you are a chess master, it might be a piece of cake to tromp.

What you are playing against, in a very real sense, is the board itself. The "brain" of the computer decision algorithm is essentially a cellular automaton (the platforms on the board). The computer's wizard marks the active cell and each cell can see the current state of all the other cells on the board (not that it uses all the information available). So, all moves are based merely on the state of the board prior to the computer's move. There is no look-ahead, no review of past moves made by the player. In short, there is no artificial intelligence in this game. It is a completely deterministic, cellular automaton designed to provide you with at least a minimal challenge (which is what I expected at first), with the minimum of coding effort.

Release Locks will remove all of your Locks from the board at a cost of one point. It is here in the off chance that you somehow manage to get so many locks on the board that you don't have any power to do anything.

The *Fast* Alternative
A HARD-DRIVE ROUNDUP

by David Plotkin

Except for a laser printer, a hard drive is probably the most expensive peripheral you will buy for your ST. Although the price is high (compared to a floppy drive), the advantages are many. Once you have tried a hard drive, you will wonder how you ever got along without one.

The first advantage is a lot of storage—hard drives range in size from a minimum of 20 megabytes (the equivalent of 55 single-sided floppies) to 100 megabytes (278 single-sided floppies). While you don't have an *unlimited* amount of space, having all of your most commonly used programs in one place is very convenient. Before I got my hard drive, I was constantly searching through my vast collection of floppies to find the program I wanted. Now, I just bring it up from my hard drive in a few moments.

The next advantage is *speed*. A hard drive loads and saves data at many times the speed of floppy drives, even relatively fast floppy drives like the Atari ST drives. The difference can result in a real increase in efficiency, as large files are loaded and saved in just a few seconds, while the same operation on a floppy may take many minutes—if the file could have been saved at all. Remember, the 1040ST and Megas can hold more data in memory than can be saved on a disk. You could, therefore, write a program which couldn't be saved on a floppy disk. I've done it myself.

Installing a hard drive

In general, installing a hard drive is much simpler on the ST than on many other computers, because the ST has a plug on the back into which you simply plug the disk drive. Special software then needs to be placed on your boot-up disk so that your ST will recognize the existence of the hard drive, and your desktop will need to be modified so that the hard drive icons appear. These operations are satisfactorily explained in most of the manuals for the various hard drives. Once the hard-drive icon is visible on the desktop, you use it just like any other drive.

In addition to the software necessary to make your ST recognize the hard drive, other pieces of software are required, and some hard drives come with additional software that makes it easier (and safer—more on this) to use the hard drive.

First of all, you need to be able to format the hard drive. This procedure is similar to formatting a floppy disk; it prepares the disk to receive data. Though all of the hard drives listed below arrive preformatted, you may need to reformat the disk if you have problems with it. Just as with a floppy, this erases all the data on the disk, so it should be done only as a last resort.

The other reason for reformatting your hard drive is to change the partitions. Let me explain. Your ST is capable of recognizing a hard drive with a maximum size of 16 megabytes. Since the smallest drive sold is 20 megabytes and the largest is five times that size, this would seem to present a problem. The solution is to "partition" the single physical hard drive into several "logical" hard drives.

For example, you could set up your 20-megabyte hard drive as two 10-megabyte hard drives, perhaps called "C" and "D." This has the added advantage of allowing you to keep similar or related programs on the same hard-drive partition. However, once the partitions have been set, the only way to change them (for example, to give yourself four 5-megabyte drives instead) is to reformat. All the drives come with formatting programs. All seem to work fairly well, although some are more sensitive to locating bad sectors on the hard drive and locking them out.

Another piece of software that is absolutely necessary is a program for backing up your hard drive to floppy disk. A disadvantage of the hard drive is that so much valuable data is kept on them that if anything goes wrong (ominously referred to as a "crash"), a lot of data and programs can be lost. Further, hard drives are sensitive to things like dust and jolts. It is recommended that your hard drive be periodically backed up on floppies, and then these floppies can be stored away against the day that the hard drive crashes.

The backup programs are different from standard copying programs, since a file to be backed up could conceivably be bigger than one floppy.

Several techniques exist for backing up hard drives, some are more convenient than others, the trade-off being speed. The backup program that comes with Astra's drives works relatively well, squeezing more information on a disk by use of a special formatting scheme, but it is slooooow, especially when the disk is unformatted and the program has to format it. David Small's *Meg-a-Minute* is considerably faster, but uses a "mirror image" technique of backup. I'm not sure what would happen if you tried to restore the data on a different drive. There are also incremental backup schemes, which are smart enough to back up only those files that have changed since the last backup.

All the hard drives listed here come with some kind of backup program except for the ICD drives—but commercial backup programs are inexpensive. There are even some excellent public-domain

(to page 53)

THE ST GAMESHELF

In the last installment of "The ST Gameshelf," I mentioned that, although there are hundreds of games available for your ST, a good many of them are rehashes of old themes. This isn't necessarily a criticism, though. Sometimes this rehashing adds a new and exciting feel to a worn-out theme.

Most of the programs we're going to look at this time around fall under the "rehash" category. As far as game themes go, there really isn't anything new here; however, as with any artist, a clever program developer can take an old idea and make it into something special.

By Clayton Walnum

Blockbuster
Mindscape, Inc.
3444 Dundee Rd.
Northbrook, IL 60062
(312) 480-7667
Color only, $39.95

A perfect example of an old theme renewed with exciting results is *Blockbuster* from Mindscape. If you're a fan of breakout-style games, then this is one program that is essential to your software library. Just the sound effects alone are worth the price!

As with most games of this type, the object of the game is to knock out all of the blocks on a screen. In order to accomplish this feat, you're supplied with a ball and a paddle. The ball bounces around the screen while you use the paddle at the bottom of the screen to keep the ball in action. Each time the ball strikes one of the multicolored blocks, the block is removed from the screen. (Some of the blocks require multiple hits.)

To add a little spice to the action, there are, randomly hidden inside the blocks, tokens that, once released, fall to the bottom of the screen. Each time you manage to grab a token with the paddle, the next higher weapon icon lights up. What weapon icons?

Bouncing the ball isn't the only way you can knock out bricks; you also have at your command (if you've gathered enough tokens to "buy" the one you want) nine different weapons that may be used to give you a little advantage. The "slowdown" weapon reduces the speed of the ball by 25 %; the "magnet" weapon allows you to catch the ball with the paddle, rather than just bounce it; the "divide" weapon places three balls on the screen simultaneously; the "wide" weapon elongates your paddle; the "torch" weapon lights up invisible bricks; the "laser" weapon adds laser firepower to your paddle; the "smart bomb" destroys all the aliens on the screen; the "missile" weapon adds another form of firepower to the paddle; and, finally, the "force field" weapon allows the ball to pass through bricks without bouncing, though it still destroys every one it touches. Whew! What an arsenal!

Aliens? Yes, I did mention aliens, didn't I? The aliens are creatures that roam around the screen, deflecting the ball and generally being a nuisance. Each time the ball hits one, it's destroyed and adds 100 points to your score. And, as I said, the aliens also deflect the ball, so you have to be careful!

The sound effects are the best I've ever heard on an ST arcade game. It's hard to describe them; that's how original they are. All the sounds are digitized, and you'll spend your first few games not trying to get a high score, but making sure you hear all the sounds.

Blockbuster also comes with a program that allows you to create your own screens. You use the mouse to place the different colored bricks, as well as to set various game attributes, such as the number of tokens that will be hidden and the number of aliens that will be released. You may add these self-created screens to the master disk.

Blockbuster is positively habit-forming. There are hours and hours of entertainment here for anyone who takes the plunge.

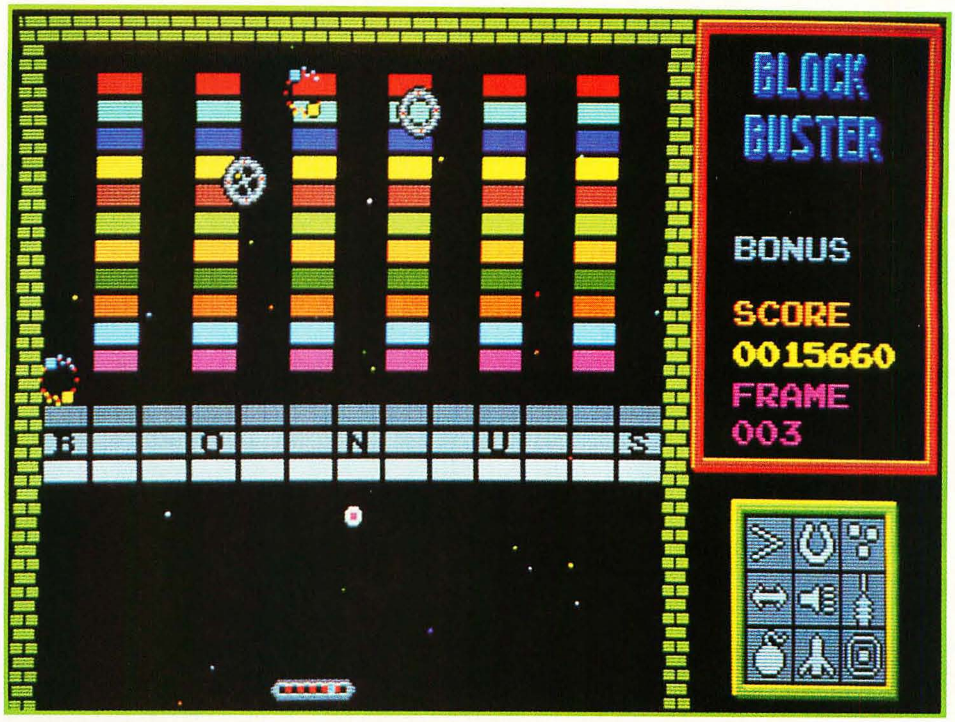
Recommendation: Buy it.

Oids
FTL Games
P.O. Box 112489
San Diego, CA 92111
Color only, $39.95

FTL had the distinction of being one of the first companies to release a game that really took advantage of the ST's graphics capabilities. That program, *Sundog*, was a smash hit, not only because it had very little competition in the ST game market, but because it was a unique and playable game. Recently FTL released *Dungeon Master* and immediately added to its reputation by being the proud publisher of what turned out to be the best- selling piece of ST software of all time.

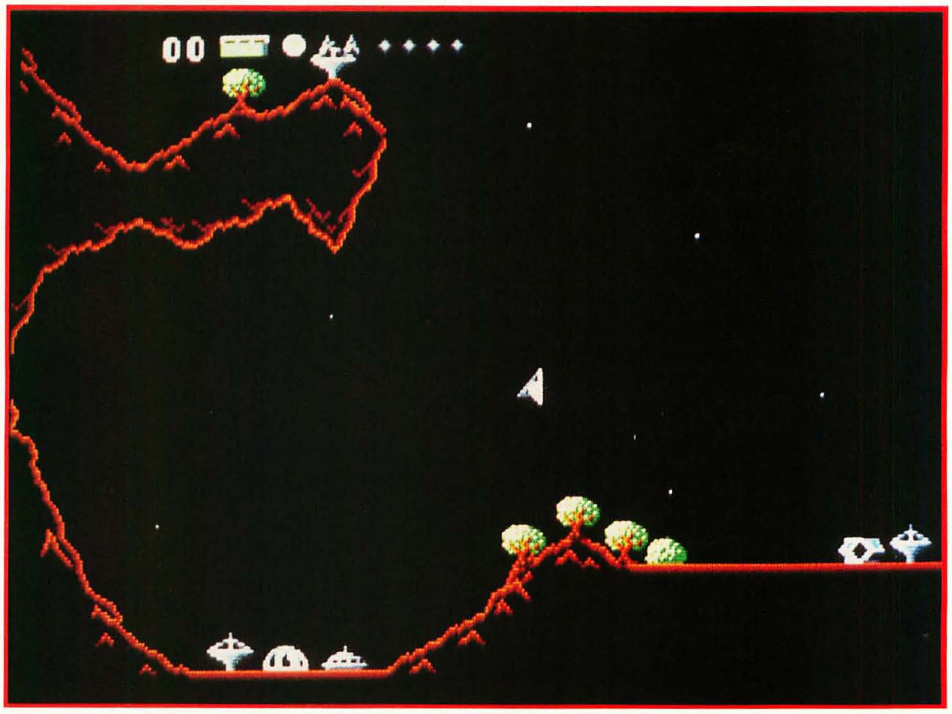
Now on the shirttails of those two sensational products comes something a little different, *Oids*. Does this game live up to the reputation that FTL has created for itself?

That's a tough question to answer because Oids is a very different type of game from Sundog and Dungeon Master, both of which are basically graphic-oriented adventure games. Oids is a kind of Lunar Lander/Choplifter combination (new ideas on old themes, remember?), which, if you wanted to stretch the term, could probably be placed in the adventure genre, though, I think it falls more easily into the arcade category. *(to page 45)*



A perfect example of an old theme renewed with exciting results is *Blockbuster* from Mindscape. If you're a fan of breakout-style games, then this is one program that is essential to your software library.

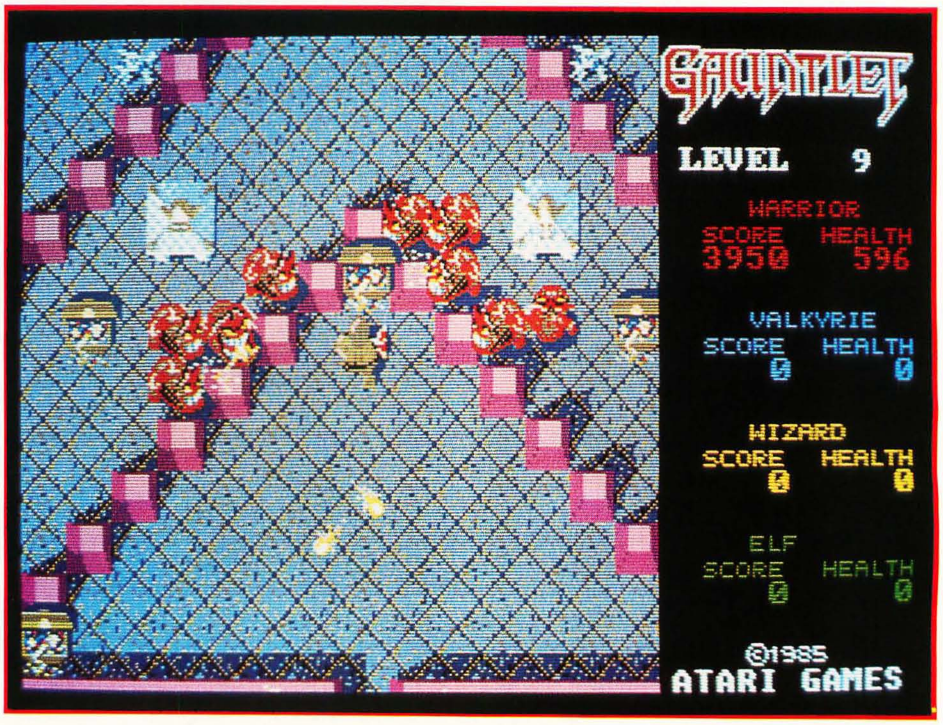
Oids is a very different type of game from Sundog and Dungeon Master, both of which are basically graphic-oriented adventure games. Oids is a kind of Lunar Lander/Choplifter combination.



One of the oldest video-game themes is the shoot-'em-up. Over the years, there have probably been more games of this type than any other, and while *Gauntlet* combines a few adventure elements to the long-standing shoot-'em-up theme, it's not enough to break it free from the shoot-'em-up category.

Gauntlet offers one- or two-player play, with the two-player mode allowing both players to work together to achieve the highest score possible. And getting a high score is about the only goal in this game. What that means is that you have to wander around each level of the dungeon maze, gathering treasures and fighting off billions of bad guys. There's food scattered about to help you keep your energy up, and the lucky adventurer will also stumble upon potions that may be used to help destroy the attacking creatures.

You fight the creatures in three ways: throwing weapons, using hand-to-hand combat or using magic. *(to page 45)*



Gauntlet offers one- or two-player play, with the two-player mode allowing both players to work together to achieve the highest score possible. And getting a high score is about the only goal in this game.

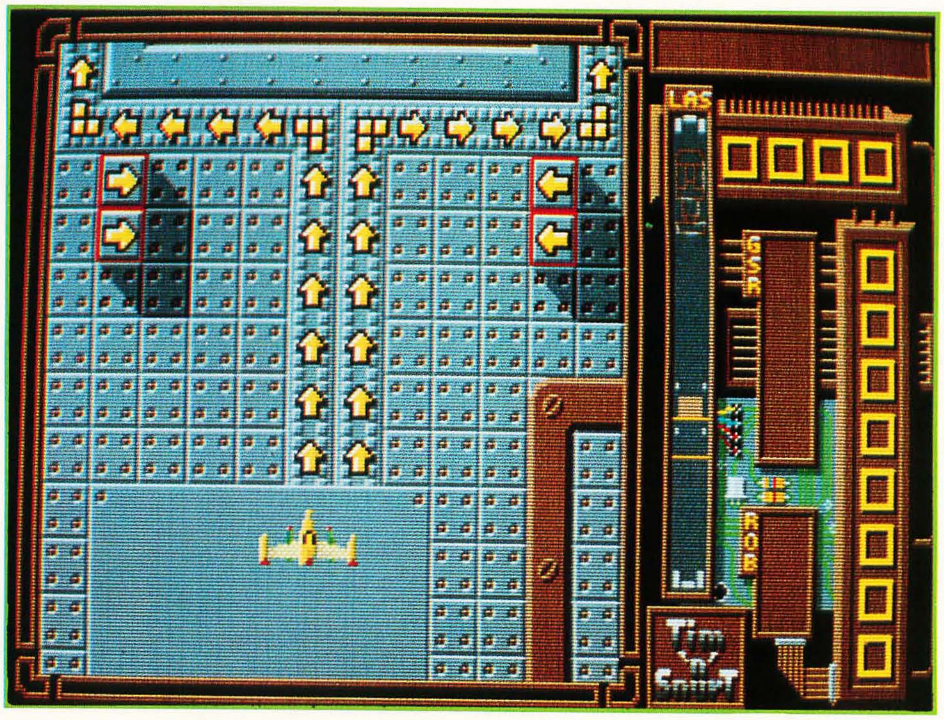
Goldrunner II
Microdeal
576 Telegraph
Pontiac, MI 48053
(313) 334-8729
Color only, $39.95

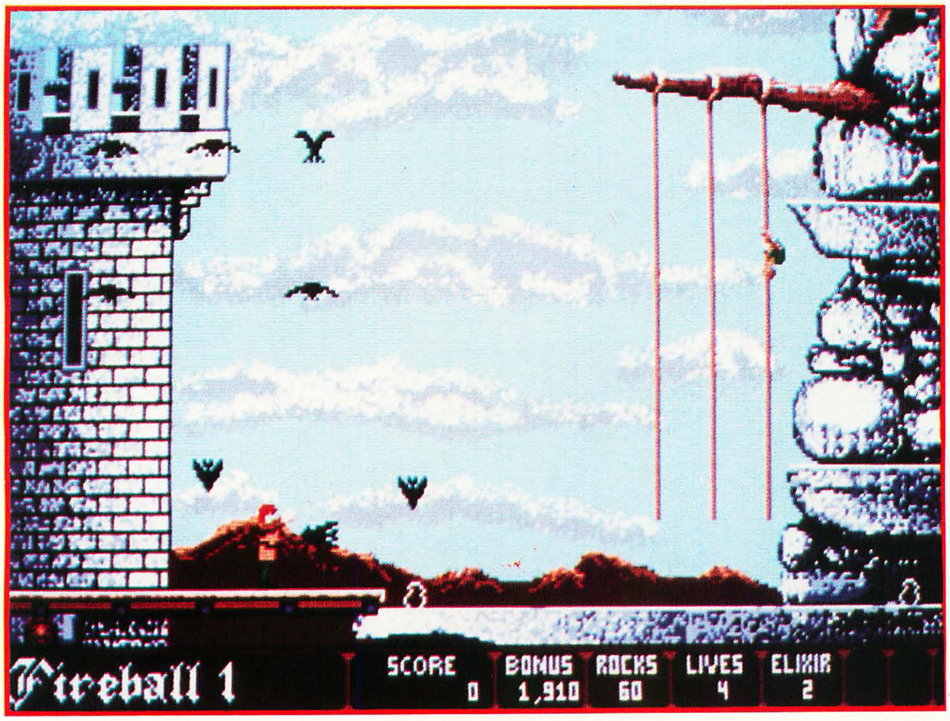
And speaking of shoot-'em-ups, I'm sure most of you remember Microdeal's successful *Goldrunner*. Goldrunner's claim to fame was its stunning graphics and great scrolling effects. Now Microdeal has released *Goldrunner II*, and I like it even more than the original.

Fans of the old Goldrunner will find that the controls work similarly; so you'll be able to get right into Goldrunner II without a lot of practice. The game play is similar, too. Though the alien ships fire upon you, they can't get through your shield. The only way to get "killed" is for your ship to collide with one of the taller structures on the scrolling terrain.

Even though the enemy's fire won't destroy your ship directly, getting hit applies a certain amount of force to your ship, causing it to move in the direction of the hit. If you get hit too many times, your ship will start moving so fast that you won't be able to avoid a crash. *(to page 46)*

Fans of the old Goldrunner will find that the controls work similarly; so you'll be able to get right into Goldrunner II without a lot of practice. The game play is similar too.





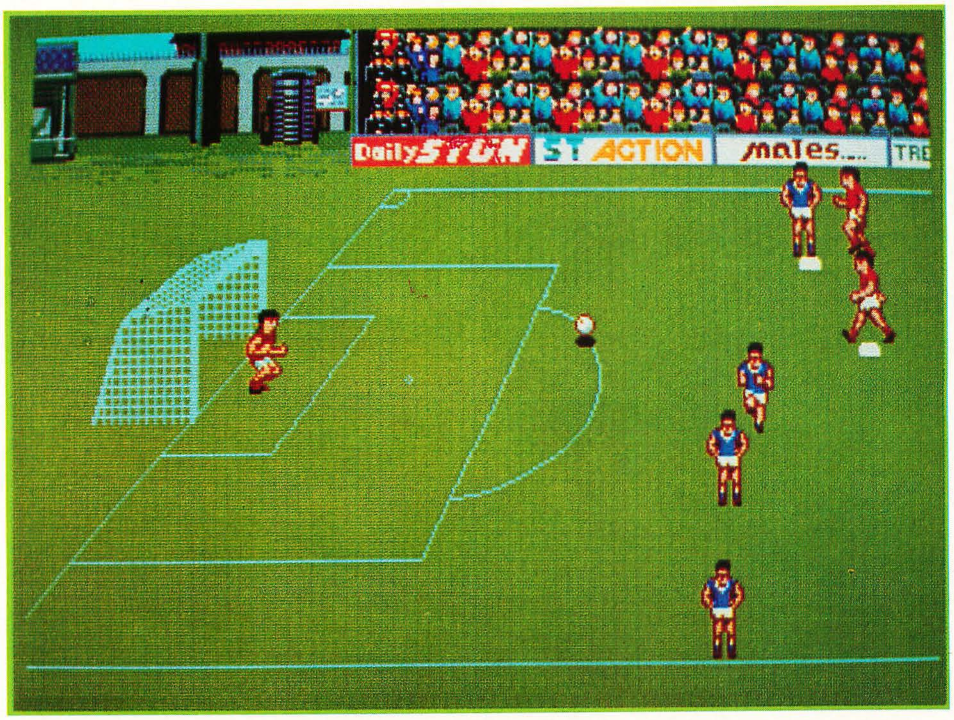
The first time you play Dark Castle, don't be surprised if your score doesn't even break 100. This is a game that requires *practice*, especially considering the unusual user interface.

Another nice offering from Microdeal is *International Soccer*, an exciting and well-executed sports simulation that you may play against the computer or against a fellow soccer fan.

Dark Castle is one of those frustrating adventure games wherein the player must find just the right combination of moves to get past one screen and into the next. Each screen of Dark Castle is a puzzle that must be solved, and while you're trying to find your way to the next level, you must do your best to stay alive by killing off rats, bats, dragons, mutants and all other sorts of nasty creatures. Your mission is to make your way through the haunted castle's 14 levels (good luck) and find and overthrow the evil Black Knight.

The first time you play Dark Castle, don't be surprised if your score doesn't even break 100. This is a game that requires *practice*, especially considering the unusual user interface. One hand must be used on the keyboard to control your movement and the other hand must be used on the mouse to throw rocks. Believe me, you're going to have master these controls before you have any chance of getting very far in this game. *(to page 46)*

International Soccer
Microdeal
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700
Color only, $39.95



Another nice offering from Microdeal is *International Soccer*, an exciting and well-executed sports simulation that you may play against the computer or against a fellow soccer fan.

The program has plenty of options that you may set to customize the game to your tastes. Included are the abilities to change the color of the player's uniforms, change each team's formation, select day or night play, select whether the wind will be blowing, set the time for each half of the game, and choose between a wet or dry field. (On a wet field players are likely to slip and fall.)

Game play is unbelievably simple for a simulation involving large teams. The computer places a white rectangle beneath your team's currently active player, usually selecting the player who is closest to the ball. This is the player you control with the joystick, though, if you don't like the computer's selection, you

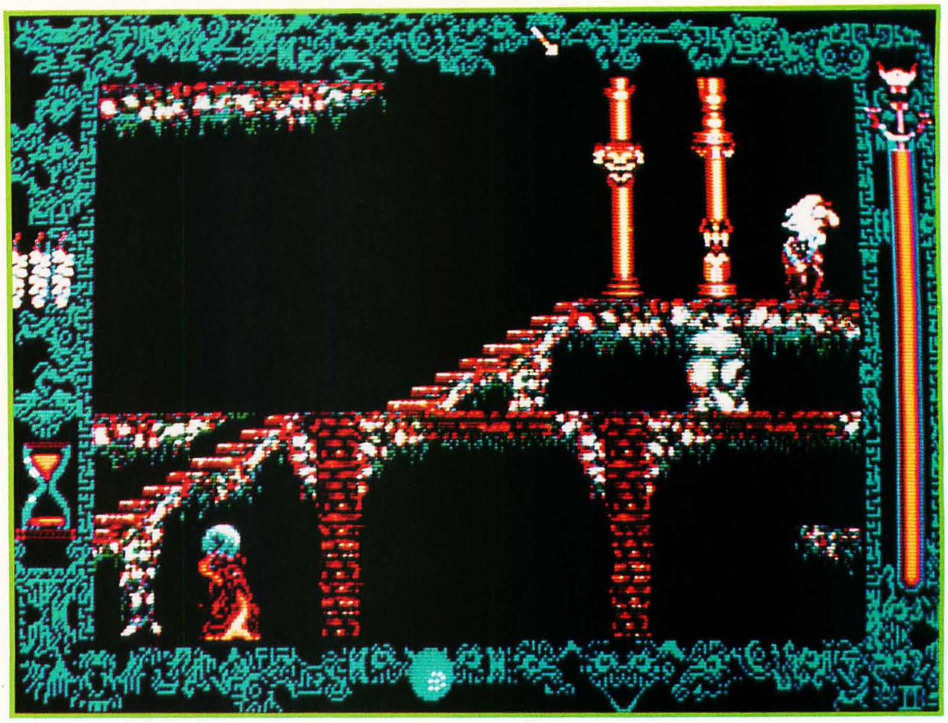
(to page 46)

Vampire's Empire
DigiTek, Inc.
10415 N. Florida Ave., Suite 410
Tampa, FL 33612
(813) 933-8023
Color only, $29.95

Okay, no pussy-footing around. I'll come right out and say it: *Vampire's Empire* has got to be one of the most difficult and frustrating games I've ever thumbed a joystick at. Unfortunately, the darn thing is so clever that you don't want to give up trying to figure it out, so you spend hours trying to survive a lot of ghoulies just so you can grab a beam of light.

That accursed beam of light! The idea, you see, is to guide a beam of light (using mirrors and a magic ball) down into the depths of Count Dracula's castle, and the beam of light is constantly in motion. Finding that ★&$%! light is tough!

The multilevel castle is loaded with trap doors that drop you down and staircases that rarely go exactly where you want them to go. Just trying to find the light beam may be more frustrating than the average gamer can tolerate. Once you do find it, you can set up a mirror to direct it deeper into the castle or catch it with the magic ball and then release it in a different direction. *(to page 47)*



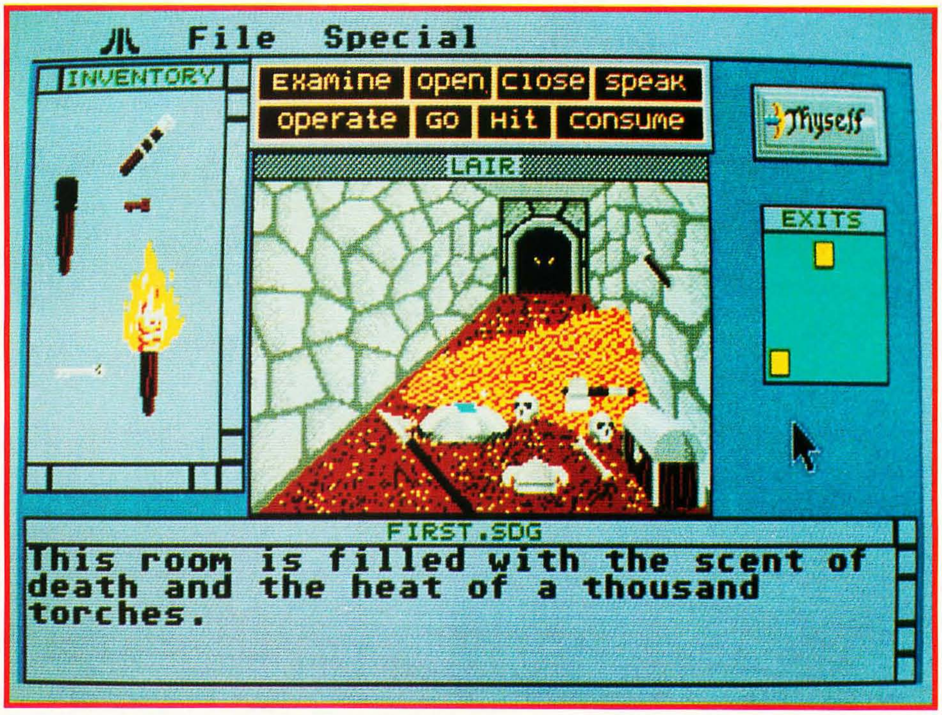
***Vampire's Empire* has got to be one of the most difficult and frustrating games I've ever thumbed a joystick at. Unfortunately, the darn thing is so clever that you don't want to give up trying to figure it out, so you spend hours trying to survive.**

Shadowgate
Mindscape
3444 Dundee Road
Northbrook, IL 60062
(312) 480-7667
Color only, $49.95

This new game from Mindscape adds a new wrinkle to the art of text/graphic adventures. Here at last is a text/graphic adventure that requires no typing! That's right: no typing. All the commands are entered by clicking on-screen commands and items. This system eliminates the time-consuming and frustrating task of trying to find just the right words to accomplish what you have in mind. With *Shadowgate,* you can concentrate on the puzzles, not the semantics.

The Shadowgate game screen is divided into four windows. These windows contain your inventory (graphically portrayed), a picture of your location, the existing exits and the game's text. The inventory and text windows, though not GEM windows, contain all the standard parts of a GEM window (scroll bars, arrows, etc.), allowing the player to manipulate them as he pleases, even to the point of having them fill the entire screen. *(to page 47)*

This new game from Mindscape has a new wrinkle to the art of text/graphic adventures. Here at last is a text/graphic adventure that requires no typing! That's right: no typing.



This room is filled with the scent of death and the heat of a thousand torches.

OIDS (from page 41)

The graphics aren't anywhere as near stunning as those of Dungeon Master, though they are adequate and perfectly suited to the game. The sound effects are reasonably good, though, not outstanding. Is Oids a step backward for FTL? Yes and no. Placed next to Dungeon Master, Oids just doesn't have the pizzazz to stand up; however, that is not due to any lack in Oids, but rather due to the fact that DungeonMaster is so sensational. Considered away from the Dungeon Master prejudice, Oids is a fine and worthwhile game that any game company would be proud to have in its catalog.

Ah, but what about the game? Basically, you have to pilot a spacecraft through a variety of alien landscapes in an effort to release the "Oids" from their slave labor in the factories and bring them to the mother ship which will take them home. All manners of alien weaponry and craft do their best to impede your progress. On the higher levels, you're in a constant battle for your life, while at the same time you must attempt to rescue the Oids.

The ship may be controlled by a joystick or the keyboard (I prefer the joystick). Just as with the lunar-lander-type programs I mentioned, you have to continually give the ship enough thrust to keep it from crashing on the planet's surface and to keep it going in the direction you want. To slow your ship requires the use of "reverse thrust," turning the ship so that the engines face the direction of travel and applying thrust to slow the ship.

As for weaponry, your ship is armed with "nuclear pellets" and "nova bombs," the former being released in a machine-gun fashion, while the latter is dropped one at a time to the planet's surface and detonated. You are also given a shield which may be used to help you get out of tough scrapes. Nothing can harm the ship while the shield is activated, but an activated shield is a continual drain on energy. When your energy gets low, you have to land and refuel—which could be a problem if you're not near a "Biocrete fuel base."

There are plenty of surprises in Oids. Things pop up from underground to fire on you, and alien ships suddenly appear from any direction. Ground weaponry can be found almost anywhere—sometimes even hanging upside down at the top of the screen.

And, like Blockbuster, Oids comes with a construction set that allows you to create your own levels. There are many new screens, created by other Oids players, available for downloading on such telecommunications services as DELPHI. In fact, FTL awards $100 to the best Oids galaxy that gets submitted to their monthly contest. The contest runs until March 31, 1989, so you've still got plenty of time to get in on it.

The first time I sat down to play Oids, I started at about 11 p.m. When I finally decided it was time to put the disk away, the sun was coming up. Need I say more?

Recommendation: Buy it.

FTL had the distinction of being one of the first campanies to release a game that really took advantage of the ST's graphics capabilities.

GAUNTLET (from page 42)

There are four different heroes you may choose among for your character, each of which has varying weaponry and magical abilities.

The graphics are well-done, though, the sound is on the chintzy side. As for challenge, the first few levels are easy to complete; things don't really get tough until around the eighth level. Unfortunately, making things tough in a game like this means putting a lot of creatures and things on the screen. When that happens, the game attains a sluggish feel as the program tries to keep up with all the animation.

Gauntlet holds very little appeal for me, though, I'll admit that there may be a lot of people out there that like this type of game. I think it gets boring fast. The animation keeps things interesting for a few games, but after a while even that begins to get humdrum.

Recommendation: For shoot-'em-up fans only.

GOLDRUNNER (from page 42)

In Goldrunner II you must recover robots that have been stolen by space pirates. To accomplish this you must first blast a series of moving ground pods (called Transporter Cars), each of which contains a robot. Blasting a pod releases a robot, but you can't pick it up. You have to wait for one of the enemy recovery ships to pick up the robot, after which you must blast the ship and then grab the robot from the air before it disintegrates.

Once you've gathered up a number of robots, you must drop them onto "teleport zones," where they will be teleported back to the Ring-Worlds of Triton. When all the robots have either been teleported or destroyed, you must then dock with the mother ship (a simple process), after which you'll advance to the next level.

Every type of meanie, including Recovery Ships, Magnet Fighters, Magnet Mines, Noumenon Missiles and various types of fighters, will be doing their best to ruin your mission, making Goldrunner II a challenging and action-packed game.

The graphics are marvelous and the game play addicting. Another fine feature is the availability of additional scenery disks. For $14.95 you can get a whole new world to explore and conquer.

Recommendation: A must-buy for shoot-'em-up fans.

DARK CASTLE (from page 43)

The graphics and animation in Dark Castle are high quality. The sound adds a lot of charm to the game, and you'll sometimes find yourself smiling. One of my favorite sounds is the one you'll hear when you hit one of the "mutants" with a rock—a squeak that brings to mind someone stepping on a mouse.

Your hero can walk in four directions: left, right, up and down—as well as jump. In order to rack up any kind of a score, you're going to have to have both hands functioning like a finely tuned machine. Don't let your hero fall off a platform. If it's just a short fall, he'll walk in circles (and make amusing groaning sounds) for a couple of seconds, dizzy from his hit on the head. During that time you're helpless and vulnerable to attack. If you fall too far, you lose a life.

The program comes on three floppy disks, and if you plan on running the program from these disks, you'd better be prepared to do a lot of waiting for disk access and to do a lot of disk swapping. Luckily the program can be copied to a hard disk or a RAMdisk, speeding up the transitions considerably. But even when not running the programs from the floppies, you must keep Disk A in Drive A; Dark Castle uses the "key disk" copy protection scheme.

The "manual" consists of one 5½-inch by 8½-inch card. Like many adventure games of its type, Dark Castle expects you to figure out the details of the game. The instructions tell you little more than the fundamentals of movement.

If you're willing to spend a lot of time practicing, you'll probably find Dark Castle to be a fun-filled challenge. If, however, you frustrate easily, you should pass this one up.

Recommendation: Get a demonstration before buying.

SOCCER (from page 43)

may select another by pressing the joystick button with the stick in the center position. As the ball moves around the field, the computer will automatically switch control between the players so that you rarely have to select a player yourself.

When one of your players gets the ball, he may then dribble the ball down the field or kick it. To dribble the ball you simply move the player, making sure you avoid contact with the other team's players. (They'll take the ball away.) Kicking is accomplished by pointing the joystick in the direction you wish to kick and then pressing and releasing the fire button. The longer you hold down the button, the harder you'll kick the ball. The ball is kicked when the button is released.

If the ball gets too close to the goal, the goalie takes over. He is controlled by pointing the joystick in the direction you want the goalie to dive and then pressing the button.

When a team is awarded a throw-in, corner kick or goal kick, the computer selects a player to do the job, then waits for you to press the space bar. While you are waiting, the players on the screen are constantly shifting position. Also supported by the simulation are penalty kicks, where you may kick the ball either high or low, depending on the position of the joystick.

Each screen of Dark Castle is a puzzle that must be solved. While you're trying to find your way to the next level, you must do your best to stay alive by killing off rats, bats, dragons, mutants and all other sorts of nasty creatures. Your mission is to make your way through the haunted castle's 14 levels and find and overthrow the evil Black Knight.

This soccer simulation is carefully engineered for ease of play. You won't have to spend much time getting used to the controls, and you'll be an old pro within your first few games. The graphics are nice and the animation smooth—another quality product from Microdeal.
Recommendation: If you like sports simulations, buy it.

VAMPIRE'S EMPIRE (from page 44)

Of course, below you there are other mirrors that may direct it back up again! *Sheesh!*

On top of all that, you are constantly attacked by bizarre creatures that come out of chests you must pass by. And that's not to mention all the stone faces (carved into the castle's walls) that come to life as you pass under them and spit some sort of awful green stuff down on your head. Each encounter with one of these unsavory creatures takes away some of your life force. You can fight back, of course. You have garlic that you may use to throw at the creatures or lay in their path; and you may also kick the nasties if they get close enough. If worse comes to worse, you can just run away, although some of these characters are tenacious, to say the least, and will follow you wherever you go.

The manual is ridiculously brief, not even taking enough time to describe the full game. For example, there's an arrow at the top of the screen that shows you where the light beam is in relation to your position. There's no mention of the arrow in the manual, and the first time I played the game, I had no idea how I was supposed to find the light. After about an hour of play, I noticed that the arrow changed direction as I moved.

The graphics are great and very unusual looking. The castle's interior seems to glow and gives the effect of wandering through darkened rooms. I just wish the game were as easy on the patience as it is on the eyes.

There appears to be a couple of minor glitches in the game. Sometimes, when falling down several levels of the castle, the figure you're controlling vanishes from the screen and doesn't reappear until the completion of the fall. Also, firing garlic at approaching monsters is a dangerous way to fight, due to the fact that the garlic only seems to actually fire about half the time.

I said before that the game's cleverness keeps you playing, that it stops you from giving up. That's true, but only to an extent. I finally gave up. I just couldn't get that beam of light to go where I wanted it to, and it seems to me that this game was targeted toward someone with a lot more patience and perseverance than I possess. I suspect that a more complete manual, one that included some hints on game play, could have made a lot of difference here. I have this feeling that I must be missing something—something that should have been in the instructions.

So Count Dracula is still alive and well in Vampire's Empire. Maybe someone out there has the patience to find a way to destroy him. As for me, I'm moving on to the next game.
Recommendation: Get a demonstration before buying.

SHADOWGATE (from page 44)

Because the text window works this way, you can review the game's text just as you would a document in a word processor, by scrolling backwards and forwards through the text.

Above the GEM-like windows is a command window that contains eight buttons labeled "Examine," "Open," "Close," "Speak," "Operate," "Go," "Hit" and "Consume." These are all the commands that are necessary to complete the adventure. But don't get the idea that this limited "vocabulary" makes the game easy. The eight commands combined with the large number of on-screen items gives you plenty to think about.

Some of the built-in commands have shortcuts. For example, double-clicking on an item (except a door) is the same as clicking on the examine button and then clicking on the item. Double-clicking on a door opens it (if it's not locked), and then double-clicking on the open door causes you to go through it.

To "operate" an item, you click on the item you want to use, click on the Operate button, and then click on the item you want to operate on. Combining two items with one command in this way yields a large number of possibilities for the adventurer to try, making this game just as challenging (though less frustrating) as a standard adventure game.

Opening items that contain other objects (for example, a leather sack), brings up another GEM-like window that will show what the opened item contains. You may move items between one window and another, placing things in and out of containers. All and all, the player/game interface is nicely thought-out and executed, making this game a pleasure to play.

If you're the jumpy type, let me give you fair warning. You never know what might leap out at you as you explore the world of Shadowgate. I might mention, for instance, the banshee that appears suddenly on the screen with an unearthly scream. Believe me, if you were falling asleep at the switch at that point, you'll be awake afterwards! (And probably digging through the medicine cabinet for tranquilizers.)

Shadowgate is a winner. Even people who are not fond of adventure games might find this one a refreshing change. I'm anxious to see more adventure games from Mindscape in this format.
Recommendation: Buy it.

That's all the games we've got for you this time around. There are many new games coming out for the ST, and if the games we've just looked at here are any indication, there's going to be a lot of exciting things in the future.

I'll see you next time.

ST-U.S.E.
THE USED PROGRAM EXCHANGE FOR YOUR ST
Trade Your Old Programs for Exciting New Titles

Buy Quality ST Programs at a Fraction of the Original price.

Over 250 Titles currently in stock
New Arrivals Daily

NOW SELLING NEW SOFTWARE

Call or write today for a free price list and membership info.

ST-USE
314 Main Street
Great Barrington, MA 01230
(413) 528-4728 9 a.m. - 5 p.m. Eastern Time
MasterCharge and VISA Accepted

CIRCLE #116 ON READER SERVICE CARD.

ANCIENT

Part 1 • A Guide to Dungeon Master

by Bob Retelle

Limping along the dank stone corridors, your fighters nursing their wounds, and your magic users' mana almost exhausted, you pause where the tunnel suddenly branches. Ahead of you, the stone walls disappear into the gloom straight as an arrow, but to your right you can see the beginning of another series of zig-zags as that corridor twists and turns.

The ominous sounding "CLICK" you heard as your party's leader stepped on a hidden switch in the stone floor, followed by the distant rattle of an iron door opening somewhere ahead can only spell danger. Suddenly, the flickering light from your waning torch catches the cruel edge of a drawn sword, and from the corner of your eye, you see a brief flicker of movement just outside of the shrinking globe of light from your torch.

With a sigh, your fighters ready their weapons as your wizard tries to cajole one last fireball from the little bit of magical mana that remains. Your priest passes around what's left of the healing potion, then picks up a rock to throw with his sling.

Is it skeletons again, with their blood-red wooden shields and flashing sharp-edged falchion swords, or some other, as yet unimagined, horror? You turn into the side corridor, tensed once again for battle. Whatever they are, they're blocking the way on your quest for your own destiny—and that of the entire world!

This of course, is *Dungeon Master* (DM), the fantasy role-playing game from Software Heaven, Inc. and FTL Games. Nearly two years in the making, DM stretches the limits of the Atari ST with its highly detailed graphics and lifelike digitized sound. Perhaps the most outstanding feature of the game, though, is its playability. It's obvious that a lot of thought and hard work went into creating an imaginary world which would be challenging, while never quite becoming impossible to cope with.

As playable as the game is, though, there are times when a little aid and guidance could come in handy to get you past a particularly difficult puzzle, or to give you a little support in getting past some of the nasty creatures which populate the dungeon. That's what these articles will do. This month, we'll present an overview of the game and some general tips on getting started and surviving while you build up your experience. Next month we'll get into some of the more specific things you may run into, and ways of handling them.

As with any game of this level of complexity, be sure to read the manual that comes with DM. The 18-page story which begins the manual sets the scene for your quest, and while it's a little misleading about the ultimate goal of the game, it will give you a feeling for what it's all about. The section on magic is very important; although while the descriptions of the various elements of magical spells do give some clues as to their use, you'll probably find them more colorful than practical.

The story

Briefly, you play the part of Theron, apprentice to the Grey Lord, the most powerful of the High Lords and the greatest wizard in the world. Your Master had sent you away on an errand to protect you, while he attempted to retrieve the legendary Power Gem from deep within the mountain. Something went wrong, and in the resulting explosion, the Grey Lord was torn apart.

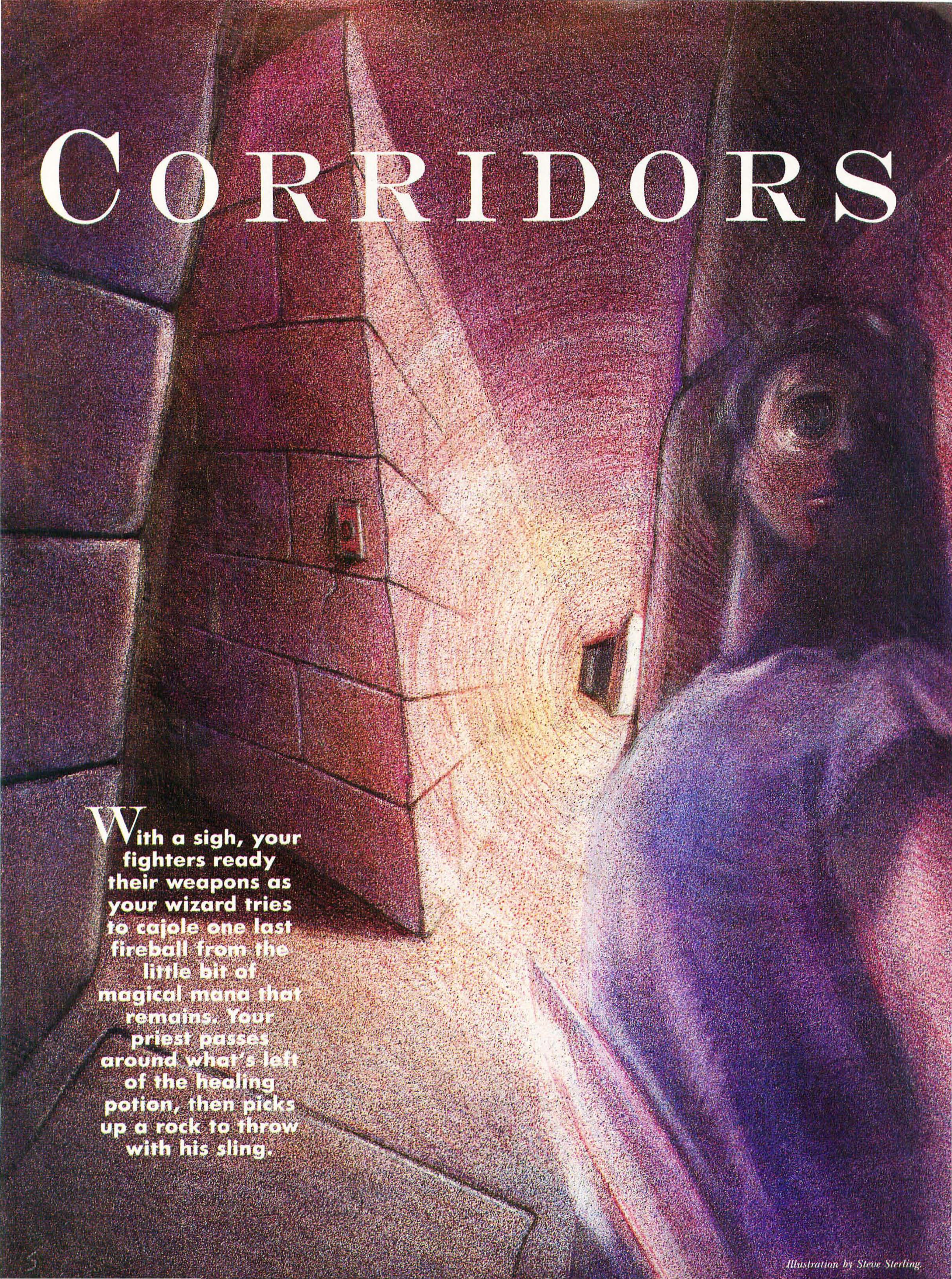
His evil side, known as Lord Chaos, locked himself in the wizard's dungeon, intending to find the Power Gem for himself and to use its power to wreak destruction on the world. The good side of the Grey Lord, unable to enter the dungeon himself, has called on you to brave the traps and monsters which Lord Chaos has filled the dungeon with, and to recover the Firestaff. Together with the Power Gem, the Firestaff can fuse the good and evil sides of the wizard together once more, and restore order and balance to the world.

Because you have been affected by the same explosion and been rendered invisible and immaterial, you must select a party of mortal champions to guide through the depths of the dungeon on this quest. You can choose from among 24 of these somewhat foolhardy individuals who had tried to brave the dangers of the dungeon without a guide—unsuccessfully. Once you have made your selection of up to four of these champions, you are ready to enter the world of *Dungeon Master.*

Choosing your champions

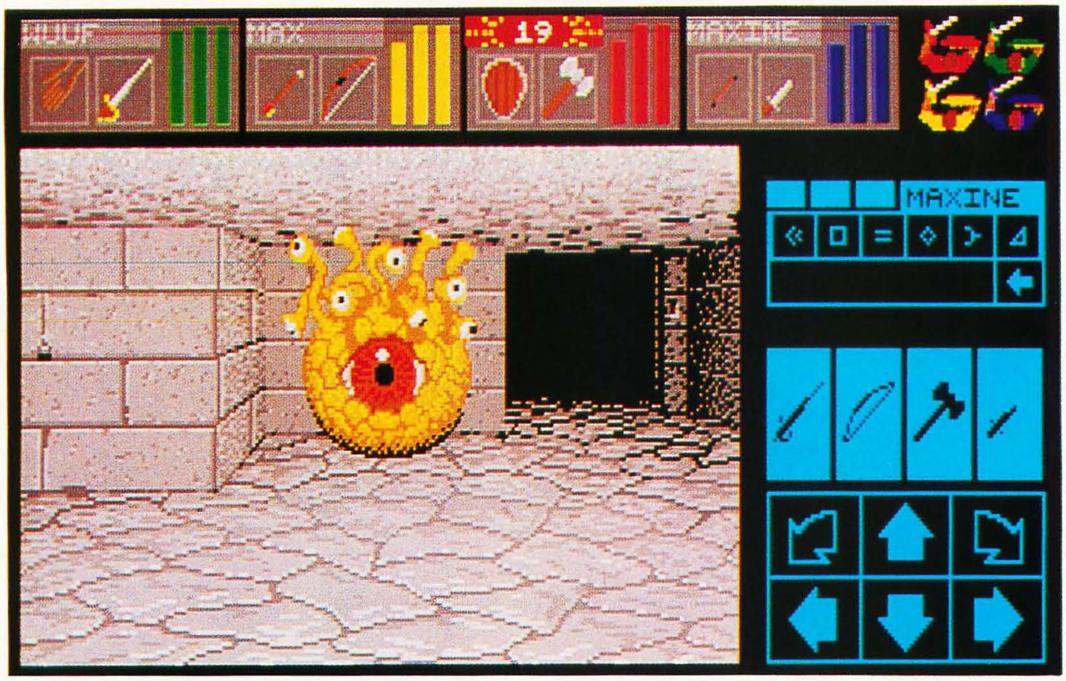
As you enter the Hall of Champions, you'll find each of the champions frozen into magical mirrors hanging on the walls.

CORRIDORS



With a sigh, your fighters ready their weapons as your wizard tries to cajole one last fireball from the little bit of magical mana that remains. Your priest passes around what's left of the healing potion, then picks up a rock to throw with his sling.

Illustration by Steve Sterling



When playing Dungeon Master, the most important thing to remember is that you are responsible for guiding your party and for looking after their needs.

Each one has the skills and equipment that he, she or it possessed at the time they fell to the horrors of the dungeon. Take your time and look at all 24 before you decide on the makeup of your party. Once you choose a champion, you can't change your mind without restarting the game.

Examine each one's skills and statistics, with the idea of forming a well-balanced party. For your first quest into the dungeon, you should probably choose two fighters with good strength and health numbers, and two with high mana and wisdom levels to be your wizard and priest. Try to choose champions who also have ninja experience, which will allow them to move quickly and aim accurately.

The numbers you'll see for each of the characteristics will change slightly from game to game, but they'll have the same general relationship. Some champions are very strong, but with very limited mana, which will restrict their ability to do magic. Others have high levels of mana, but with such low health that they will be very vulnerable in battle.

You'll have the option of choosing to either resurrect the champions just as you see them, with the same name and all the same skills and attributes, or you can reincarnate them as a new player, giving them a new name. This second option will convert their skills into slightly higher numbers for their physical characteristics, but leave them totally without any learned skills. The increase in their attributes is so slight that for your first game, it's probably better to let them keep their skills and names, and just resurrect them.

Arranging your party

Once you've chosen your group of four champions, you may find it necessary to rearrange them in order to put the best fighters in front of the group. Each of the characters has a color associated with him. The four icons in the upper righthand corner of the screen each correspond to the champion with that color. By clicking on one of the icons and moving it onto another of the icons, you can make those two champions trade places in the group. The party always walks "two by two," two in front, two in back. You should give the fighters in front the best swords and axes, while the two in back can use bows or slings to shoot over their heads.

One thing to watch about the arrangement of the group is that a person on the right will throw or shoot down the right-hand side of the corridor, and the person on the left will aim down the left side. If a monster is in the middle of the hallway, it won't make any difference, and both will be able to hit it. Often though, you'll find the monsters staying to one side or the other, especially when they've just come around a corner. This means that you have to watch carefully to see that you don't waste your arrows or fireballs by shooting down the wrong side of the tunnel. You can fire one shot, then trade places in the group to get off another shot, if you have enough time.

Playing the game

The most important thing to remember while playing the game is that you are responsible for guiding your party and for looking after their needs. You must remember to feed them and give them water. During battles, you must remember to watch over them so that their health doesn't fall too low while you're occupied choosing weapons and fighting monsters. You must clothe them, and most importantly, see to their training in the skills they'll need to survive the dungeon.

Your job as guide to the champions truly begins when your leader steps on the floor plate that opens the iron-grate door into the dungeon. Beyond that door you'll need all your cunning and bravery to guide your party to victory over Chaos.

Scrolls and altars

On the Hall of Champions level, you'll find some tasty morsels of food to put in your champions' packs, as well as some useful items like torches. You'll also find your first scrolls. In DM, scrolls are all "informative"; that is, you read a scroll to find out something useful, but the act of reading a scroll doesn't do anything "magical." To read a scroll, go to a character's inventory screen and either put it into their "action hand" or hold it over the eye icon and hold the left mouse button down. Once you've read a scroll and written down what it said, you don't need to keep it.

You'll also find the first of several "Altars of VI," where you can bring back the bones of a dead champion and have them resurrected. Thankfully, DM provides this means of recovering from mistakes or overwhelming attacks. The game is not over as long as at least one of your party survives to bring the bones of the others back to an altar.

Of course, like everything else, resurrection does not come without a price. Every time a champion is resurrected at an altar, he will lose some health points. Also, it may be extremely tedious to have to retrace your steps to the closest altar to bring a member of the party back to life, and to go back and recover all the items they were carrying when they died. In other words, try not to get anyone killed!

Food and water

You'll find food scattered here and there throughout the dungeon, but you'll need to ration it wisely if you want to avoid watching your people starve to death. Different kinds of food are more or less filling than others. You'll soon learn which foods are good for a quick snack and which can be counted on to fill a hungry champion after a long day's exploring.

One thing that cannot be overstressed, however, is *don't* overfeed your party! Wait until their food indicator drops below half before you feed them anything. They won't begin to suffer the effects of hunger until their food indicator turns yellow, and feeding them too often will only waste your limited supply of food.

Unlike food, the "Lions Head" water fountains on the walls appear to be unlimited, so you can give your champions a drink wherever you pass a fountain. There are areas in the dungeon which have no water fountains at all, though, so be sure that you're carrying a good supply. When you're in those areas, the same advice applies to water as to feeding the party.

It's not immediately apparent, but you can use an empty flask to hold water as well as magic potions. A flask will almost completely quench a very thirsty champion, while it takes three swallows from a waterskin. A flask cannot be shared among the members of the party, while a waterskin can be passed around until its three swallows are gone. You fill either a flask or waterskin by holding it against a water fountain on the wall and clicking the left mouse button.

Training your champions

The most valuable thing you can do for your champions is to train them in the various skills they need. Any of the champions can learn to use any of the skills, no matter how they may start out. Fighters may be low in magical ability and wizards may be weaklings, but by practicing diligently, they can advance to high levels.

It's important to remember to keep practicing these skills, even when there's no apparent need to be swinging swords or throwing things. If the only time you use a skill is in combat with monsters, your champions' abilities will grow slowly. Very early in the game, you will find the scroll which tells how to create a magical torch for light. If you have your fighters cast this spell as often as they can, not only will it save your normal torches, but you'll soon find them advancing to higher wizard levels, which also gives them more mana.

Use the lowest magic-power level at first. If you're told that someone needs more practice in using the spell, keep at it. They'll receive experience credit for trying. If a champion doesn't have enough mana to cast the entire spell at once, just click on the "syllables" they *can* do and wait until they regain enough mana to do the rest. It can be a long process, but it's beneficial in the end.

The "War Cry" option given to an unarmed character will give that person experience toward advancing in priest levels, which will also increase their total mana. Anytime it looks as if your party will be safe from attack for a time, you can have them put their weapons away and have a chorus of war cries to build up priestly experience. It can be nice to have each champion able to create their own healing potions and to be able to heal the others in the party when they begin to encounter the really nasty creatures deep in the dungeon!

Ninja skills affect your champions' strength and dexterity and help determine how effective they are when using "range" weapons like a bow or sling, or when throwing things. A good way to gain ninja experience is to throw objects whenever you have the chance. Scrolls, torches, unneeded pieces of clothing, anything that can be picked up and thrown out ahead of the party will give the character who is currently the party leader extra ninja experience.

The advice about giving your strongest fighters the best weapons and putting them out in front of the party generally should be followed. They'll be the ones on whom most of the responsibility for close-in, hand-to-hand fighting will rest, and they'll need all the experience they can get. There are a few places, however, where it's fairly safe to put the weaker members of the party up in front, to let them get some fighter experience too.

ProCopy **ST BACKUP UTILITY**

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

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

Dealer Inquiries Welcome

VISA MasterCard and C.O.D. orders

Call (800) 843-1223

$34.95

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

PROCO PRODUCTS

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

Available in Europe

THE MOVING FINGER CO.
Building 2
Shamrock Quay
Southampton, SO1-1QL
England
Tel. 0703-229041

TECH-SOFT
COMPUTER WHOLESALE
324 Stirling Highway
Claremont
Western Australia 6010
Tel. 09-385-1885

CIRCLE #117 ON READER SERVICE CARD.

Remember that every time a champion advances a level of experience, no matter what skill is involved, their health and stamina increases, along with one or more of their other statistics. One thing that can be said about the dungeon with complete assurance is that no matter how bad things look now, they can only get worse, and the better prepared they are, the longer your party will survive!

Notes and maps

There are several things which will aid you in playing DM and which can make life a little easier (and a little longer!). Be sure to make notes as you go along, especially when you find magical scrolls scattered here and there in the dungeon. It doesn't matter if you write them down in ancient runes or not, but the spells and clues contained in the scrolls are crucial to the game. Also, recording where you found valuable items will help you if you have to (or want to) start over.

One of the most valuable things you can do is to make a map of the dungeon as you go along. There are many twists and turns, and after a while all the tunnels begin to look alike. Knowing which corridors are safe to retreat into, and which are fatal dead ends just may save your champions' lives! Also, there are a lot of side passages which contain valuable equipment and clues that you might miss if you don't note them on a map.

Unfortunately, mapping the dungeon levels tends to slow down the play of the game and requires tremendous patience. Because DM runs in "real-time," the monsters won't wait while you draw your map, and your torches will continue to burn down while you add little details.

If you can, draw your own map as you go along; but if you'd rather not take the time, there are several sets of excellent maps which have been drawn by other brave souls. These can be downloaded from information services such as the ST-Log SIG on DELPHI and from local bulletin-board systems. If you have a local Atari user group, check to see if they have the DM maps in their club library.

Some players may feel that using such pre-drawn maps is cheating, so you should be reassured that the game can be won without mapping or using ready-made maps. It may just take a bit more concentration to remember where you are, and that you may miss some areas.

Details! Details!

The Dungeon is rich with detail, and the smallest thing may hold great importance. Your powers of observation will be taxed to the extreme. Tiny little chinks between stones in the walls may be secret switches which will trigger events, both good and bad. Gray iron keys lying on the gray stones of the floor will defy you to notice them, but notice them you must if you are to proceed. A crack in the wall may look just like a hundred other cracks, but look closely, and you may notice a tiny hidden button to press.

While you're examining the walls for hidden switches, be sure to also watch where you're putting your feet. There happen to be many floor plates in the hallways which may do things like open or close pits in front of you or slam doors behind you. Sometimes you may find it necessary to drop something onto a floor plate to keep it depressed while you go on beyond it. Things like burned-out torches and scrolls that you've already read are useful for this purpose, and you won't have to worry about leaving something useful behind.

Further down in the dungeon, you'll encounter tiny floor plates which are too small to be held down by chests or shields or in any other way than by standing on them. To keep these switches shut, you'll have to lure someone or something onto them, then use magic to hold them there.

Puzzlements

Besides the locked doors for which you'll have to find keys, and the secret doors in the walls which are opened by pressing hidden switches, you'll also encounter many devious puzzles which Lord Chaos has placed in your way in an effort to foil your quest. Most of the puzzles are quite logical, and with many you may find clues inscribed on the walls nearby. If you seem to be having trouble solving a puzzle, look around for these clues. Be observant of the areas nearby too. If you push a switch in the wall and nothing seems to happen, look around and see if anything is different from before.

The farther down into the dungeon you go, the more devious these puzzles become. You'll also begin to encounter buttons which open doors far across the level. If you find a door which can't be opened immediately, watch for buttons later on which don't seem to do anything. You may find that, if you go back, the door will be open!

Some of the puzzles can only be solved by quick action, and your party must move as fast as possible to get through these areas. Watch the line at the bottom of the inventory screen for each champion, which tells how much each is carrying and the maximum amount each can carry. If this line turns yellow, it means that the person's load is at or above half of the maximum. This not only wears down their stamina, but it makes them move slower as well. In most cases, this really won't affect them badly; they'll just have to stop and rest more often. There may actually be times when this situation is unavoidable—for instance, when you have a lot of items to carry, and you can't distribute the load among the group any more evenly.

There are several places, though, where the need for fast movement means that you may have to lighten the load so that none of your party is "in the yellow," even if this means dropping valuable equipment. Fortunately, these areas are all located so that you'll be able to recover anything you have to drop, so you don't have to worry about losing anything permanently.

To get through the puzzles that require fast movement, first drop any equipment that may not be immediately useful. Extra food and torches, spare weapons and armor can be left behind to lighten the load. Check to be sure that none of the party has a LOAD indicator which is yellow. Then put your best ninja in the lead by clicking on his/her name at the top of the screen. Remember, the leader's name is highlighted in yellow. You can check each champion's ninja skill levels by going to his inventory screen and holding the left mouse button down while pointing to the eye icon.

The last and possibly most important part of moving quickly is to *use the keyboard cursor keys to move.* Pressing the arrow keys on the keyboard is *much* quicker than clicking the mouse on the direction arrow icons on the screen! Count how many spaces you'll have to move, if possible, then with the mouse pointer, activate the puzzle and quickly press the appropriate keyboard arrow key the number of times counted.

With a little practice, you should be able to get through these puzzles without too much trouble. If you do get hopelessly stuck at any of the puzzles, fear not! Next month's article will go into more detail, with hints for specific situations—and, of course, we'll begin to talk about those charming monsters you'll encounter!

Bob Retelle has been a professional in the field of telecommunications for the past several years (look for BOBR online) and has been writing (and playing) computer adventure games for longer than he'd like to admit.

(from page 39)

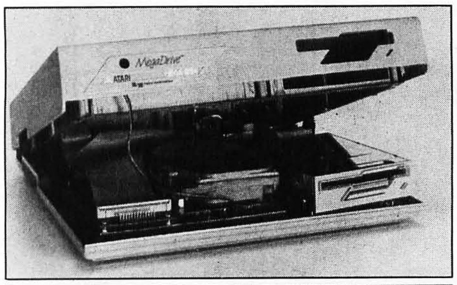
backup programs. Backing up your hard drive is such a drawn-out process that most people never bother—much to their chagrin when (not if) their drives crash.

"Parking" the hard-drive head refers to moving the head over a portion of the drive where no data is stored and locking it there. This is absolutely essential if you intend to move the drive, as otherwise you may damage some data. All the hard drives either come with parking programs or are self-parking.

The ST can also be taught to boot up from the hard drive, which is much faster than using a floppy, especially if you use a lot of desk accessories or GDOS fonts. The Supra and ICD drives come with auto-boot software, and these programs seem to work on other makes of hard drive as well. Both programs also allow you to disable the hard-drive boot by pressing certain key combinations. This is important if the hard-drive fails to boot for some reason, or if you want to create a boot floppy with a special custom configuration for occasional use without disabling the hard-drive boot and then rebooting.

The following hard drives are available for your ST:

SupraDrive
20 Mb—$699.95
30 Mb—$795.00
60 Mb—$1,295.00
Mega internal 40-Mb drive—$995.00
Supra Corporation
1133 Commercial Way
Albany, OR 97321

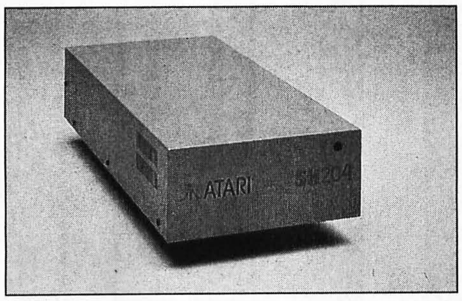


The Supra line of drives have stood the test of time well. They come with auto-boot software. The Mega hard drive fits inside a Mega ST, though it requires a dealer or skilled technician to install it.

Brainstorm—$849.95
Brain Storm
911 E Pike Suite 325
Seattle, WA 98112

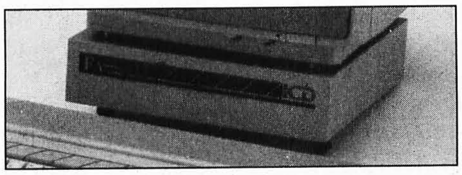
This combination box includes a 30-Mb hard drive, 5.25-inch PC-type floppy or 3.5-inch floppy, four AC outlets on the back with four switches on the front (surge protected) and a real-time clock. The box is configured to be large enough to support a monitor and is elevated so that the ST can slide under it when not in use. It comes with software to automatically set the computer clock when you boot up.

Atari SH204—$699.95
Atari Corporation
1196 Borregas Avenue
Sunnyvale, CA 94088



The Atari drive is a solid (if somewhat noisy) 20-megabyte hard disk. The front panel includes a power light and busy light.

ICD ST Hard Drive System
20 Mb—$699.95
30 Mb—$949.95
50 Mb—$1,099.95
dual 20 Mb—$1,149.95
dual 30 Mb—$1,349.95
dual 50 Mb—$1,699.95
ICD
1220 Rock Street
Rockford, IL 61101



These hard drives are in cases which are long, low and designed to fit under your monitor or Mega box. A battery-backed clock is included, along with the software to automatically set the time in your computer from the clock. An expansion port in the back allows plugging in other DMA devices (including another hard drive). The drives can be ordered in two-drive configurations (40, 60 or 100 Mb), either giving extra storage or allowing the use of one of the independent drives as a backup for the other.

Astra System HD+
20 Mb
30 Mb—$995.00
40 Mb
Astra Systems
2500 S. Fairview, Unit L
Santa Ana, CA 92704

Affectionately referred to as "the Tank," the HD+ is very rugged. It has the On/Off switch in the front, which is a welcome change. It also has a built-in 3.5-inch double-sided drive, which can be used as drive A or B (but must be drive B in a two-floppy system). Astra doesn't use suggested retail prices, but the estimated cost of the 30-Mb version is as indicated.

Home/Office Series Expander
20 Mb (expandable to 60 Mb)
30 Mb (expandable to 90 Mb)—without floppy $1,199, with floppy $1,299, extra drive units (for expansion) $400
40 Mb (expandable to 120 Mb)

This new line from Astra, available with or without the 3.5-inch floppy, is designed to fit under a monitor or Mega. It has four power outlets on the back, with EFI/EMI noise suppression and surge suppression built in. Two power switches are available on the front of the case—one for the hard drive, the other for all the AC connections—making it possible to first switch on the hard drive, then turn on the rest of the system.

RM60/120
60 Mb—$1,500
120 Mb—$2,500

Astra's line of rack-mounted hard drives is primarily designed for studio musicians. They are very rugged and expandable.

IAN'S QUEST

by Ian Chadwick

The other day I was sharing a few pints of home brew with a couple of old friends, both of whom have been computer users since the late 1970s. We were reminiscing about the "good old days" back when we first got our machines. Tom and I began with the TRS-80, back in 1978. Bill and I went on to Atari 400s, then 800s in 1983; Tom went for the Apple II about the same time.

Today, Bill and I have STs and PCs, Tom an Atari XE. We three bought our computers for much the same reason: to play and eventually design intellectually challenging games.

Tom and I started with 16K of RAM—considered to be an enormous amount of memory, since the computer had only recently evolved from 4K. The competition, the Commodore PET, had only 8K at the time. There was also a built-in BASIC which meant all of this memory was available for programming. A lot of amazing software was designed to run in a mere 16K, a lot of which has not been improved upon in megabyte-size systems. I was among the first users in my neighborhood to purchase the notoriously unreliable TRS-80 expansion box, which added another 48K RAM; 64K was simply staggering. Why, you could put a whole universe in there!

Aside from learning to program, we played a lot of games in those days. There weren't many arcade-style games around, although a company called Big Five Software produced some truly delightful efforts, especially given the limited graphics capability of the TRS-80. Most of what we played was written in BASIC, which meant we could usually break into the program and hack away at the code when we wanted to tinker. And we tinkered a lot. We whiled away hours discussing the effect of changing a particular variable, of adding subroutines and of altering code. Some of our proposed changes sent us to the library for days of research just to prove a point. We had access to the program source code, so we applied ourselves to mastering it. In the process, we learned a lot about programming, simulations and games. There were few games we didn't alter in one way or another.

By far, the majority of the games that we played and enjoyed were simulations, more or less, in game form. Most made little, if any, effort to include graphics or sound and were generally text-only display and input. And they occupied us for hours and hours of competition, play and enjoyment. Among these were *Santa Paravia*, a game about ruling a medieval kingdom; *Taipan*, a trading game in the 19th Century Orient; and *Colony Omega*, about building and running a space colony. There were games about running businesses, managing a sawmill, guiding tankers through the straits of Valdez, simulations about pest control, rat populations and buffalo herds. I loved them all. I used to subscribe to a magazine called, *Softside*, which printed the BASIC listings of these and similar games or simulations. For years, it was my favorite magazine. It was also one of the first publications where I got an article printed.

By today's standards, they appear boring, even simplistic. We've come to expect slick mouse or joystick input, fanciful sound effects, stunning graphics. The thought of a simple, text-only game with limited keyboard input fails to stir the imagination of many users who were introduced to computers through more state-of-the-art designs. Sure, they were stripped down to the bare essentials, but a lot of these games were more involving and enjoyable than many of the more recent efforts I've encountered. They demanded thought, planning and strategy.

Graphics and sound effects are chrome. Many developers and designers use them to mask an essentially dull game. Take a look at *Defender of the Crown*: lots of amazing graphics, nice sound, but essentially a pretty skimpy game underneath. The joust sequence consists solely of attempting to position the cursor (the tip of your lance) in the middle of a moving target (the opponent's shield). Ho-hum. The sword-fight sequence, when you attempt to rescue a maiden from her kidnapper, is a push-me-pull-you game requiring less skill or talent than perseverance in the mouse-clicking department. The economic sub-system is limited solely to buying and allocating weapons or soldiers. The strategic game consists of plodding about the country trying to grab the few territories before the computer grabs them all, something I've failed to do. Somehow, with about the same number of territories, the computer manages to field an army of some hundreds of men to my 20 or 30. Of course, I get beaten every time.

The idea of Defender of the Crown and the equally uninteresting game, *SDI*, is to create games with movielike scenarios which culminate in a defined plot ending. Sounds good, but they put a lot more effort into the chrome than the meat of the game. The marginally connected segments are too simple and too limited to provide long-term appeal. Instead of a coherent game, you get a hodge-podge of bits and pieces, none of which is particularly interesting.

Not to say, of course, that every game needs to be an intellectual roustabout, but its longevity depends on being able to provide continuing levels of interest and challenge. Straightforward arcade games may sell well initially, but like TV sitcoms, they quickly lose their audience. How many people would buy *Asteroids* again? Or *Lunar Lander*? *Frogger*? *Missile Command*?

Okay, sure, when the first two were combined and dressed up again in *Oids*, a lot of us bought it. But would you have bought them by themselves without the enhancements Oids gives them? Didn't think so. Think about all those shoot-em-up games you bought in the 8-bit world that nowadays you only feel disdain for.

The challenge of racking up just another 1,000 points in *Goldrunner*, or reaching just another level in *Sentry*, fades after a while. It's not much of an accomplishment to engrave your name in the high-scores list on your disk. Unless you're a serious couch potato, you can't play these games indefinitely. They are, in the long run, boring. They don't teach and you don't learn from them.

A few hours entertainment is a low return for the price: Up here in Canada, the average game sells for about $50. Even at $30, it's hard to justify buying a lot of variations on the same cosmic-froggy-space-zapper theme. The market has only a certain capacity. On the other hand, a game like *Empire* offers a lot of bang for the buck, so to speak. I've probably spent more hours playing Empire than any other game for the ST. Bill, who bought the game on my recommendation, has become an Empire addict, and it's become difficult to pry him from the computer when he's playing it. It's not really a simulation in the same sense as those I mentioned, but it does make the gray matter work.

The simulations we played in the old days were usually written by people like us; hackers, tinkerers, self-taught programmers. It seemed that, if you had a computer, you learned to program. Maybe you didn't write programs, but you learned to change other people's code. Today, despite an abundance of languages, including some excellent offerings like GFA BASIC, there seem to be fewer hackers. The percentage of users who program appears much smaller than it once did. More people seem to be passive users than active. In discussions in computer stores, my question "Do you program?" creates a response similar to that as if I asked, "Do you dive for pearls?" especially among the younger users who see the machine as a toy rather than a tool. There is a disturbing perception among a lot of people of the ST as a game machine. I'd rather see it as a learning tool.

That's too bad, really. BASIC is an easy language to learn and very satisfying to work in because an interpreted language provides instant user-gratification: The results of your efforts are seen immediately. A two-step language like C or Pascal demands that you write code using a text editor, then compile and link it before you see anything. I personally dislike this process. Besides, GFA's compiler adds the advantage of being able to create fast, machine-language code from your finished efforts.

So what's the point? It's a call for submissions, for more effort in the area of simulation. I'd like to see a lot more of this sort of game grace the pages of ST-Log. I think readers learn a lot more by hacking away at a simulation than they do trying to alter an arcade-style effort that has little, if any, real leeway for change. Adding a single variable to simulate the effects of a disease on a stand of trees to *Timber Baron* had a wide-ranging impact that altered play completely. You just can't make that sort of impact in *Droid* or something similar. And the impetus to hack—and therefore learn—is lessened knowing you can't make significant alterations without seriously rewriting the code.

Back in the Jurassic period, around the time when I was learning to use my new Atari 800, I worked for Canadian Press, selling a high-level online historical news database. You could search through news as recent as yesterday for specific topics of interest, for people, for companies, for words that might appear in a story.

I learned that an enormous volume of news never sees the light of print, due to restrictions on space or interest. Who cares about the government of Andorra? Or about a new dinosaur find in Montana? About the fiscal deficit in Zaire? The development of a new type of copy-proof gambling chip? I did and still do. It was fascinating to read the hitherto unknown news from all over the world. Since it all interested me, I spent hours on the system, which sold in those days for an online fee of $90 an hour. As the salesman, it was free to me, of course, and I took advantage of it to increase my knowledge, whenever possible. But the system was a trifle rich for my blood once I left the company.

Now, a similar system is available in *ST XPress.* XPress is a data feed with news, sports, finance, stocks, weather, recipes, quizzes, gossip, lifestyles and other material, transmitted over your television cable. A lot of this material is never printed in your local newspaper or gets airtime on the abysmally truncated news the radio and TV present.

A special decoder hooks up to your ST's serial port. You must subscribe to the data service and either buy or lease the decoder in order to use XPress. The rates are very reasonable for the value received. The software to capture the data and select material from the feed was written by Alan Page, co-author of *Flash,* my favorite telecommunications program.

Press software was originally available for the IBM PC, but when Alan took on the job of writing the ST version, he decided to make several improvements. As a result, ST XPress is considerably more flexible and powerful than its PC cousin.

XPress isn't a historical system: It presents the up-to-the-minute news, with stock and commodity data, from the Standard and Poor's Portfolio, delayed by a mere 15 minutes. Stories are saved in memory and can be recalled or saved to disk. There's a lot of news that comes through and many stories are duplicated in a day, so Alan's software provides the ability to selectively limit what you receive by selecting only certain categories to save. There's also a keyword search through stories in memory, to help further in finding items of interest from stories in the chosen categories.

By far, the most powerful feature is the clipping folders. These permit saving stories using Boolean logic operators, OR, AND and NOT. The default is OR, but you can specify stories in which only certain words appear together (AND) or in which words do not appear. Clipping folders even work on incoming stories outside your chosen categories.

You can save stories in memory or clipping folders to disk as ASCII files; you can print stories or even load clipping folders from disk. Unfortunately, there is no Autosave feature.

Securities—stocks, commodities, exchange indexes—don't automatically display in ticker-tape fashion. You'd soon overload on data. Instead, you enter the security symbol you want to monitor, say ATC for Atari Corp., AAPL for Apple Corp. and CBU for Commodore (the symbol legend is provided regularly through the feed or hardcopy can be obtained from Standard and Poor's). You can add various suffixes, such as *C* for Canadian or *X* for mutual funds, where applicable. Only data for those symbols you enter are displayed.

Alan, being one of the rare breed of programmers who understands the need for keyboard commands to supplement the GEM-mouse structure, provides a wide range of Alt-key combinations which perform most of the functions in the menus.

Finally, XPress can work in background mode, so you can be capturing news stories while you're using another program. Alan tested several programs and found that many—including DEGAS, *A-Calc, Thunder, Fleet Street Publisher, ST Writer* and *WordPerfect*—work with ST XPress. Programs that demand most of memory or use the serial port won't work.

The value of ST XPress should be immediately apparent to anyone interested in current events, sports or stocks. It should also interest researchers, teachers and parents. What a gold mine it is for kids needing background material for a school project! I urge you to look into ST XPress—it's a new and different application, to my mind the most important new use for a computer to come along in several years.

Ian Chadwick is a technical writer and editor living in Toronto. He is the author of Mapping the Atari, *the 8-bit reference guide, published by Compute! Books. He is currently writing a computer simulation to "put his money where his mouth is." He can be reached on DELPHI by sending mail to username CHADWICK.*

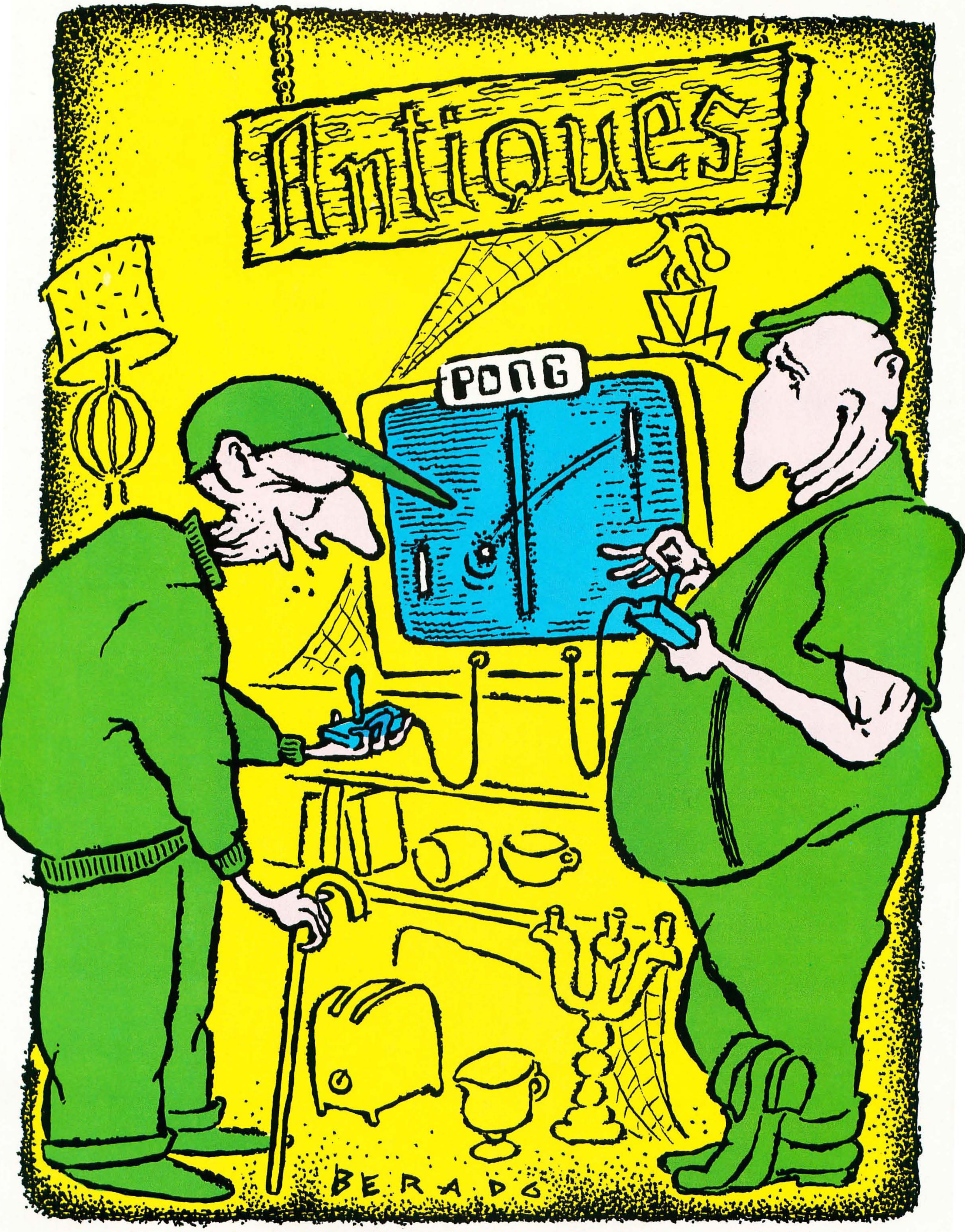


Illustration by John Berado

The Birth of

A Partial List of Features

Fully interactive 2 dimensional & TRUE 3 dimensional capabilities
Multiple 3D views can be opened and modified at any time
A user can work in any combination of views with all views instantly updating at all times
Automatic generation of ANY orthographic view including user defined auxillary views
Entities can be selectively hidden in any view allowing easy generation of true orthographically sound views
Extremely user friendly
Full GEM interface, pull down menus, mouse or keyboard commands, dialog boxes and more.
256 Layers can be activated invidually or in groups
9 zoom modes allow magnification's of up to 1,000,000 times
The ability to overlay other programmes from within DynaCADD

Dimensioning

Auto Dimensioning includes;
Mechanical and Architectural formats
Full 2D and 3D dimensioning is supported
Baseline, chaining, angular and linear dimensioning is supported
Automatic tolerancing in any of three different styles
Text orientation using any one of the three different systems (uni-directional, angled or aligned)

Commands

Sophisticated command nesting allows the following partial list of commands to be accessed at any time:
Zoom in or out
Zoom a window
Scroll or "Pan" the page
Center the page on a point
Multiple 3 Dimensional dynamic rotations at any angular increment
Rotate any 3 Dimensional view to a predefined co-ordinate plane
Zoom a 3 Dimensional view in or out…

Fonts

Full featured Vector Font Editor
Automatic proportional spacing can be activated or deactivated with the click of a mouse
Up to 16 extremely high resolution fonts can be active in any drawing with no need for repeated disk access

Insertion

Insert POINTS, LINES, CIRCLES, ARCS, FILLETS, ELLIPSES, ELLEPTICAL ARCS, TEXT, BOX, POLYGON, POLYFIGURE and SUBFIGURES using a wide variety of modifiers

a New Breed

DynaCADD™ Computer Aided Design and Drafting

Introducing DynaCADD™...a Professional CADD Solution at a Fraction of the Cost

DynaCADD is a 2D and *true* 3D Computer Aided Design and Drafting software solution that will turn your personal computer into a sophisticated and powerful CADD work station. With electrical, mechanical, architectural or civil applications DynaCADD provides a total design and drafting solution. DynaCADD revises, designs and details drawings in 2D or true 3D, reads and writes the industry standard DXF file formats and provides compatibility with the new Desktop Publishing solution, *Calamus.* DynaCADD proves itself invaluable from conception to output by offering an alternative solution to expensive, difficult to use 2D and 3D CADD systems. DynaCADD brings mainframe CADD capabilities to the CADD Professional at a fraction of the cost.

SPECIAL OFFER!!

For a limited time only
Save up to $150.00 off our introductory price
Trade in your existing Atari ST CAD package and receive up to $150.00 in credit against the purchase of DynaCADD

Call us for details

Information Commands

DynaCADD gives you the ability to:
Measure distances, angles and perimeters
Verify location and style
List database extents, drawing parameters...

Entity Transformation

Translation between 2D or 3D positions, circular arrays, mirror, linear arrays, stretch, sweep, rescale, project, revolve, trim and divide
3 Dimensional entity transformation of any view into 2 Dimensional entities
Select entities for transformation & editing using;
Digitize, Windows, Windows Out, View Windows or Last...
By entity color
By entity layer
By chaining with direction and much more

Plotter Output

Both HPGL and DM/PL and compatible plotters are supported
Plots can be generated at a constant 1:1 ratio or a drawing can be automatically scaled to any degree
Plotting extents can be defined using:
The drawing page
The current extents
A definable plotting window
The database extents
Plotter and printer output can be directed through the serial port, parallel port or to a disk file (ie; i/o redirection)

Printer Output

Laser Printers, draft and final mode
Epson and compatible printers are supported, both 8 and 24 pin
All drivers allow;
Quickplots, Final drafts utilizing the printers highest graphics mode, multi sheet prints, scaled and constant ratio prints

Data Transfer

DynaCADD reads and writes both DXF Entity and DXF Drawing Files
DEF File Format is included to insure upward compatibility to all versions on all micro computers
DynaCADD allows compatibility with the new Desktop Publishing solution, Calamus

System Requirements:

Atari 1040ST or Mega series, minimum 1 MByte of memory, 1 MByte disk drive and a Monochrome Monitor

Commitment to the Future

A modular design structure provides the basis for continued upward growth of the DynaCADD solution. A strong commitment to research and development and a world wide support mechanism combined with a young experienced dynamic programming team, are your insurance that the DynaCADD package will remain on the leading edge of CADD technology. Upgrade modules currently under development include surface design and image design, both with the same easy to use graphic interface of the DynaCADD base module.

To order and for more information please contact:
ISD Marketing, Inc. 2651 John Street, Unit #3, Markham, Ontario, Canada L3R 2W5
Tel: (416) 479-1880 Fax: (416) 479-1882
CIRCLE #118 ON READER SERVICE CARD.

T
F

WO

UTURES

TOMORROW

WHO'S GAME FOR FUTURE GAMING?

by DANIEL SILVESTRI

If we play electronic games, we assure the publishers that we will wear the games out, that our appetite will demand more sophisticated games in the future, and that the publisher can have a long-term profitable business if they listen to us, the consumers, and respond to technological changes. Here's to the electronic games, past, present and future! And here's to Atari who I know will outpace the competition with their machines!

The history of electronic gaming is riddled with a myriad of games that have been No. 1 sellers and No. 1 losers. Occasionally, games will withstand the test of time (the true test of greatness in most things), but often they fall prey to advancing technology. Electronic gaming, more than anything else, is independent of most generic tests or ways of measuring greatness. Why? Because emerging technology dictates that we, as human beings questing for knowledge and for betterment, want to keep up with the times. After all, that has been how the human race has advanced over millions of years. . . .

Technological stream flows one way

We have all played great board games, and perhaps still play board games, for amusement and challenge. Board games exist in every category imaginable. The difference between great board games and great electronic games is that we do not expect the great board game to get better or to be replaced with a better version. So, some board games have the luxury of enjoying prosperity literally for decades upon decades. Games like *Monopoly* and *Clue* from Parker Brothers sold well 20 years ago and sell well today. Their packages have changed, but the games remain the same, intact, for years and years.

Electronic games are very different. How many of us would run out and buy a *Pong*-like game today? An electronic hit just five years ago, Pong has been sold and resold in garage sales over the last several years! Why? We read, understand and demand more! We are, for the most part, educated electronic consumers who understand that technology has advanced around and past those games. We do not want to play with old technology; that's the beauty of technology, isn't it? We want manufacturers to push their development teams to discover newer, faster, better ways of doing things. When they do, the new is old, and on we go.

We have gone from *Pong* to *Pirates* (by MicroProse), and technology is a moving, changing stream that flows only one way: upward and forward.

The teeming stream of gaming

As game fanatics, we have probably played every type of electronic game we can think of. Basically, outside of recreating electronic versions of board games, electronic games are divided into four categories: (1) Arcade games which demand heavy hand-eye coordination; (2) arcade/action games which demand hand-eye skills plus some strategic thinking to complete the games; (3) historical or simulation games, much like the games published by Strategic Simulations; and finally, (4) adventure games that are either text-only games or those which combine text and graphics, which can include role-playing games as well.

Arcade games are still extremely popular because they afford long hours of playing time, new challenges with increasingly hard "levels" of play, and so on. Games like *Pac-man, Space Invaders* and hundreds of similar games have been extremely successful because of this. These games have a decent life expectancy, but oftentimes, you will find them in a dusty drawer or in some disk case that has not seen the light of day for quite some time.

The movement towards combining arcade/action games with some strategy has met with success as well and appeals to a wider target market: those who like arcade/action games have this available to them here, and those who like to have to think out some strategy have that option as well. The life of these games is generally shorter than that of arcade/action games because there is a sense of accomplishment when we "complete" the game, and often we feel it necessary to move on to something else.

Historical or simulation games stretch the strategy portion of the game and stress thoughtful play, while any type of arcade/action takes a backseat. These games have a long life expectancy, as we do not expect the "Battle of Shiloh" to be any different two years from now than it was over 100 years ago. It is a simulation and as such, we expect less of it over time and are therefore more tolerable of it later on. We might play these games over and over again, and over a period of many years, with huge gaps in between.

Finally, the adventure game is perhaps the most popular and most disposable game to hit the market. There are a variety of types of adventure games: text-only, text mixed with graphics, and animated graphics with text (covers a broad range of topics). All of them allow the player to "solve" the game, and once solved, the player will probably not go back to play that game again. These games have a life expectancy of several months, at which point they are generally solved and stored in your library of software.

A quick docking at the software library

How many of us have a software library? I do. Now how many of us have a library or section where we keep all of our books? Most of us, like me, do. The fundamental difference between our book library and our software library is technology. We have low expectations of our books. We know that they will not change over time, and that a good book, like a good board game, can be picked up again after five or ten years and reread without offending our level of expectancy.

Not so with our software library. I have many, many games in my software library, but I always want to play new ones which exploit new technologies. Forget the adventure games which I have solved; they're solved, and as such, they are like a final exam: When its over, you don't look back! For the most part, my software library card does not have very many entries for old technology. Check yours out. My software librarian has been released due to lack of interest. We want *new technology*, or we'd still predominantly be playing board games, which are in a safe harbor from technological waves. This just further supports my opening paragraph: If publishers will listen to us and technology, they will do well, and we will get what we demand.

What lies up ahead on the software stream

From what I see, the biggest changes are going to take place in the adventure gaming portion of the business. Arcade/action games and simulations will get better, as will arcade/strategy games; but the advancement will not be as monumental, since the nature of these games does not demand monumental leaps.

Several concepts will be addressed in future electronic games: (1) Games will become more realistic, that is, they will reflect how things are done in the real world, and similar actions, strategies, experiences will become more and more apparent on our terminal screens; (2) they will become more feature-rich as memory expands and graphic capabilities continue to grow; this will lend itself to the realism discussed above; (3) story lines will be expanded to match the capabilities of our new technology; the trend to write stories by teams of writers, graphic artists and programmers will continue and expand; the new games will be massive, detailed and rich; and (4) enhanced graphics will become an integral part of solving the game, not just an unacceptable substitution for your imagination.

Demand for text-only games will begin to soften, as all of the above begins and continues to happen. Personally, I have had tremendous fun with text-only games, and I plan to continue purchasing them. However, I think as we enhance our systems more and more, we will demand greater satisfaction. Major game publishers like *Infocom*, who have traditionally said that they will *not* do graphic-oriented games because graphics, no matter how good, are no match for the imagination, have announced that they will produce games within the year that take advantage of graphics. Wow! This is a far cry from "sticking our graphics where the sun don't shine," which was their banner! I fully expect them to implement graphics in creatively new ways and to make the graphic games be as close a match to their fantastic imaginations as possible.

The trend towards enhanced graphics is clear not only in gaming software but in business software as well. Graphics are now a large part of all business-application software sold. If it's getting bigger in business, you know it will get bigger in the gaming world, where graphic creativity can run amuck!

Look for it, it *will* happen!

What I would like to see on the stream

Well, I am all for advancing civilization and for pushing the envelope of technology. Now that my humanitarian statement is completed, I would like to say that I am all for seeing our gaming world change.

It must change, because electronic gaming is only part of our technological revolution. Advancing computer science demands that segments of this industry change or become history. I am surprised in a way at Infocom, yet in another way, very happy to see that it is *responding* to the technological environment, rather than *reacting*. Reacting generally implies it's too late, while responding implies thoughtful planning. It, like many other companies, is positioning itself for the future. *Bravo!* I have always admired bright, creative, innovative companies like Infocom, and I respect the attempt to move up the technological stream!

I have made a wish list of some ideas that, given the changing technology itself we have been discussing, might be implemented into future games.

(1). How about documentation that really enhances the game, and sets the tone and mood for the game? So much more can be done on this level, that any move in this direction would be fantastic. By creating a detailed background story, for example, the game, when played, becomes even more enhanced. We understand where we begin in the game, what has "happened" in the past, why the adventure is necessary and so on. Creating the right ambiance can add new vim and vigor to the games. Naturally, the greater our technological achievement, the more detailed the story line can be. Documentation should keep pace.

(2). Let me know what you think, but I would like to *play* games, and not write novelettes of notes or become a cartographer because I need to make more maps than Rand-McNally. I think the increased capacity of machines in memory and graphics ought to steer the publishers in the right direction. Write these routines in the games; if we have to take notes, give us electronic notebooks in the game; if we need a map because the terrain is confusing, draw it for us! Infocom's *Beyond Zork* draws the maps for you and that has been my absolute favorite game from it; after all, map-making is not a strategic talent for electronic games—please, do it for us!

(3). Role-playing games, or games where you spend a good deal of time developing a character or a band of characters have tremendous entertainment value. A couple of years ago I got my two nephews started off on an adventure that lasted a good year for them. The bottom line is, these games are fun, but require a great deal of time and creative energy to develop such characters. What about being able to move these characters from adventure to adventure, even from publisher to publisher? So many games have similar story lines that it would be natural for my Ultima characters to take off on an adventure in another realm—perhaps into space! I think through enhanced import/export capabilities, this can be accomplished among a variety of publishers.

(4). Finally, I would like to see more games that employ support materials that are pertinent to the play of the game. Infocom has always been great at this, and they are getting better. Other publishers have begun using these techniques as well. This approach accomplishes two things: It embellishes and enhances the story line and game playability; and it affords some protection to uncopyprotected games in that it discourages the software piracy which hurts us all.

Conclusion

As we move up the stream of technology, we will see many of these things implemented, and many more that minds more creative than mine will dream up. High-tech advances on the hardware side will always drive the software market harder and harder. That means great opportunities for us. Publishers can listen to their dedicated consumers and respond to technology, and we can play more sophisticated and advanced games.

So as you all look through your software library one more time, make sure that you clear enough space on the shelves for what is certain to come: electronic gaming software that now can only be dreamed of. Soon we will all wake up to this new reality!

T W O F U T U R E S

TOMORROW'S TOMORROW

THE FUTURE OF COMPUTER GAMING

BY MICHAEL A. BANKS

Remember when you thought *Pong* was the ultimate in computer gaming? Or, maybe the first time you played *Adventure* you figured nothing could beat *that*. Now the state of the art is exemplified by *Flight Simulator* and *Ultima IV* and similar games. It's all pretty amazing when you think about it: from *Pong* to *Lode Runner* and *King's Quest III* in less than a decade. But what's next? Where is computer gaming going?

As always, better displays and faster hardware loom on the horizon, along with programming tricks to smooth out animation and better simulate human reactions and real-world situations through increasingly complex approaches to artificial intelligence. But this is nothing more than evolution and refinement of existing hardware and programming tools and techniques.

To put it another way, gaming in 1989 involves the same activities as those in 1979: manipulating dots on a screen with a joystick and "talking" through a keyboard. We do these things faster and slicker and with increasingly complex images and scenarios, but the basic ideas are the same.

The truth is, gaming development has reached a plateau in terms of what can be done with the types of equipment we have. We can only do the same things better and faster. Game designers have been able to improve their tools only marginally over the past few years, and spend more time concentrating on using their imaginations to feed the gamer's imagination with increasingly realistic scenarios. (The latter is not all bad by the way; in fact, it's something I may address in another article.)

Not convinced? Consider this: In one respect, *Flight Simulator* differs from *Pong* only in that there are better-looking dots—and more of them—directed by more complex data files. And *King's Quest III* differs from the original *Adventure* by virtue of increased complexity and the addition of moving dots.

Has computer gaming, then, reached maturity? Will games, like movies, be limited to minor technological flourishes and rely solely on the imagination of their creators for new elements?

I think not.

Gaming has always been the hottest element of the personal computer marketplace, and that's not going to change. Because of this, game publishers will continue to support the development of more original games based on existing technology. But games have always moved to take advantage of new technology, too—better monitors and the like. And it is from new technology that the really startling changes in computer gaming are going to come over the next few years.

When I say "new technology," I mean new directions and applications (although totally *new* technology will have some effects too). Specifically, the computer games of the future are going to depend more on the evolution of input and output devices—and I'm not talking better color monitors and joysticks. What is needed—and what will be developed—are totally new approaches to exchanging information with a computer. Some will be possible only with faster processing and greater amounts of memory (both system memory and storage).

And what might those be? Here are some thoughts.

Input (player-to-computer)

Footpads. (already on the market) These use pressure sensors to convey information about the player's foot and body movements and position to a program. (These have been used with skiing games; move your feet as if you were wearing skis to control the movements of the character on the screen.)

"Wands." These are similar to the "electronic drumsticks" now on the market. But instead of being wired for sound, as it were, these devices provide motion feedback to game software. They could simulate swords, ski poles or any of a number of other objects. (I'm sure you're thinking of several I haven't named as you read this.)

Devices to detect skin conductivity and temperature, respiration and heart rate. This data could be used by the software to infer

your emotional state. In an action/arcade game or simulation, the software might switch to a different track of action calculated to deliver greater challenges when you reach a certain state of excitement—or when you relax overly much.

Motion sensors. Imagine wearing a lightweight helmet that tracks the motion and angle of your head, or, equip that helmet with sensors that detect eye-muscle movements. Now, zap that information to the computer, so it has even more information about how you're reacting to what it's displaying—or even which section of the computer screen you're looking at!

Or, slip on expanded versions of the cuffs used with blood-pressure testing devices, and let the computer "feel' and respond to your arm, leg or torso movements. Where hand and finger motions are required, use "waldos" (gloves with micro-sensors, invented by Robert A. Heinlein in 1941 for use by a victim of *myasthenia gravis* in his novelette, *Waldo*). Add to this a footpad and a wand to serve as a sword—or two to serve as ski poles—and the implications are obvious.

Voice input. Voice-response technology is still in the infancy of its development, and its problems are many (variations in speech patterns from one individual to another, as well as variations in an individual's speech patterns from day to day). Still, progress is being made, and some experimental systems are turning in impressive results. But even primitive systems can detect whistles, the difference between high- and low-pitched voice tones, or the difference between single- or multiple-syllable words—any of which could deliver simple commands without worry about word recognition or sentence parsing.

Output (computer-to-player)

At this point, personal-computer game output is limited to what can be displayed on a computer monitor and piped to a speaker. Those aren't insignificant, but imagine a computer game that could give you these:

Motion. This seems a fairly tall order—after all, how do you simulate, say, the sensation of rapid forward acceleration, or the sideways force of a race car rounding a turn? Easy, and it's already been done, using what are called physical cues, or slight motions which, when accompanied by proper visual and/or audio cues, are amplified by a receptive mind to fool the body into thinking greater motion than is really being experienced is taking place.

If you've ever "flown" a sophisticated flight simulator, you know how this works. Given the proper visual display and a cooperative imagination, and someone who is tilted back in a chair a mere five degrees and pushed slightly forward will feel that he is undergoing some heavy vertical acceleration. Physical cues have even been used in amusement park rides and educational displays. (Try the Space Shuttle "Flight" at the Alabama Space & Rocket Center if you get the chance.)

Tactile sense. The same devices used to sense motions could supply tactile cues by applying pressure to certain parts of your body. For instance such a device might pinch your arm slightly when your RPG character is injured during a battle. Small, though the real sensation is, this kind of cue could easily be translated into a sharper pain by a receptive mind.

Smell. Okay, this gets complicated, as smells tend to linger in the air. Still, small jets of air could carry appropriate scents past your nose at appropriate times, followed by neutralizing jets of clear air (or by a special neutralizing scent).

Taste. This is even trickier than smell, but there are possibilities. A gadget might be devised that could squirt or spray a flavored liquid into your mouth, but the problem is almost too difficult to consider (and taste lingers more than smell). If this aspect of sensory input ever gets off the ground, it might be in the form of something along the lines of "scratch and taste" cards, perhaps supplemented by scent cues.

Sight and sound can be enhanced as well. High-resolution, large-screen projection displays are in sight, and 3-D television and projection holograms offer some promise in the long run.

For sound, a true stereo system utilizing speakers or headphones to deliver the next generation of music and speech synthesis would be ideal. (The current state of the art in speech synthesis is truly amazing, by the way. One major communications company is currently using a system that reproduces *accents*; look for some astounding developments in this area for personal computers in the near future.)

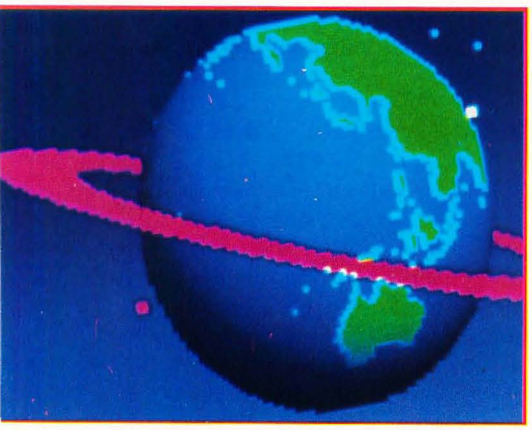
It's all possible. . . but where is it?

Interestingly enough, most of the foregoing is possible using existing technology. Indeed, much of it is in use or being experimented with by defense and aerospace interests. (For a glimpse of what's not yet possible, but definitely probable, see the accompanying sidebar.) However, there are several reasons why we aren't seeing much in the way of such innovations in the computer marketplace as yet. The most important of these has to do with the way computer game publishers market their wares. As long as the sales of products using existing technology are good, most publishers are not going to actively support new technology. And with no support for technological innovations, those innovations are slow to come to the marketplace.

However, a few publishers are beginning to utilize new wrinkles in technology. Epyx's venture into VCR gaming is a good example of what I feel is the leading edge of a trend toward new technology in gaming. (Of course, we will see a lot of VCR and other technology brought to gaming, but I won't go beyond what will happen with computer gaming here.) As game software publishers realize there's money to be made by supporting or even introducing new technology (and they are realizing it), new technology will hit the marketplace.

Too, there is precedent for the better-known computer manufacturers to introduce new input/output technology of such import that software publishers immediately move to support it, if for no other reason than fear of losing something to the competition. Apple's introduction of the mouse with the Lisa and Macintosh is a good example of this—and look what happened. And why do computer manufacturers introduce hardware innovations? To remain competitive.

The bottom line is this: As software publishers see new technology in use by their competitors, and as computer manufacturers with clout introduce new input/output devices, gadgetry such as I've described here will show up in the marketplace.



Another reason we don't yet have access to, say, body motion sensors, is the fact that developing such equipment is *expensive.* However, any product becomes less expensive when either or both of two things happen: demand makes large-scale, lower-cost production possible, and/or technology decreases the cost.

A final logjam in the development of these new input/output devices is "ease of use," which has no little bearing on market acceptance. Obviously, putting compressed air-powered cuffs or pasting electrodes on your body is rather inconvenient when all you want to do is play a game. As the technology for less obtrusive devices develops, they become easier to use, and market resistance diminishes.

Game design

Now we come to the part that I know many of you have been waiting for: my ideas as to what *kinds* of games the future has in store for us. That is the easiest part of all this, because the trend that leads to the game category of the near future is right in front of you, in every major game" software publisher's catalog.

I'll pause now for you to consider the nature of this revelation.

(Pauses.)

Okay, if you thought I was going to say "simulation," you're right. It's a clear trend; mindless alien-blasting is on the way out. Simulation of real-world adventure, exploration and encounters are here to stay. We'll continue to see air-, space-, land- and sea-vehicle simulations (battle and otherwise), but the field will expand to include activities as diverse as mountain climbing and rappelling, caving and more—all with ever-increasing realism. We'll travel to other countries, a la *Where in the World is Carmen Sandiego,* either on adventurous quests or to learn—or both. History enthusiasts will relive famous battles, or carry on dialogues with noted thinkers of the past. College students will "try out" careers.

And that's just the tip of the cliched iceberg.

How do I know this is the way things are going? As I said, it's a clear trend. Aside from that, it *has* to happen. Anyone who has been a computer gamer for more than a few months is beyond target shooting or talking with programs that have all the personality of answering machines. Games that replicate what's already available are going to die in the incredibly competitive marketplace, the demographics of which are dominated by novelty seekers who want *realism.*

This is not to say that fantasy adventures or simulations are out. Everyone wants to escape—that's why science fiction remains popular in all medias. But just as other entertainment media have matured in content (from Chaplin to Eddie Murphy in film, from Chuck Berry to Phil Collins in popular music, and from Hawthorne to Heinlein in speculative literature), so must games mature and reflect the times.

Thus, simulations are the future of gaming. Whether the venue is a Tolkienesque fantasy or a chillingly realistic encounter with death, tomorrow's computer games will simulate reality or pseudoreality, as the case may be.

To wrap this up, I'm going to hazard a few predictions, anyway. And "hazard" is the appropriate word, indeed. Prediction is a risky business at best because reality is always skewed when compared with expectations. It's safe enough to talk about trends, but assigning definite time lines to definite events is usually a mistake. Still. . . .

Where will you put it all? Access and storage

In case you haven't realized it by now, a game that uses the kinds of input and output just described will require relatively massive amounts of computing power and storage. The computing power is on the way—it already exists, in large part—but storage is another matter entirely. Obviously, the program and data files for such a game will be large. Fast access will be necessary too.

The memory system that can provide both massive storage and fast access is CD-ROM. As yet unimagined innovations may be just around the corner, but I think it's a safe bet that CD-ROM will be with us for several decades at least; too many "mover-and-shaker" corporations have too much invested for it to be otherwise. And we're all going to buy CD-ROM units eventually, which means a massive installed user base to perpetuate CD-ROM technology.

So, count on games moving gradually to CD-ROM as a media of necessity—and convenience.

The near future (1989-1995)

Despite the appeal of multipurpose machines, I think computers are going to be discreet entities for a long time, and that means we can expect to see the developments I've discussed become realities for computers as well as for arcade and dedicated gaming machines. Computers may be combined with other electronic devices, but they will retain their identity.

Okay, on with the program. We'll see a lot of hardware experiments over the next few years, some of which will turn out to be silly failures and some of which will be accepted and, thus, open for later evolution. VCR/computer links will be pushed as a new frontier in graphics, but the access time of such systems will never rival those of CD-ROM systems, and VCRs will fall by the wayside as computer peripherals. Among the major computer-gaming peripherals will be footpads, which will be used in flight simulation programs (rudder pedals) and other simulations such as skiing.

Look for simple motion sensors in the early 1990s. These will probably take the form of devices that transmit the movement of a player's hands to a computer via FM or infrared to provide a wireless analog to the joystick. These could be wands, or strap-on gadgets, and will transmit to a base unit that will plug into a computer's joystick port.

At least one game will come equipped with skin-conductivity and temperature sensors. I can't predict what the game will be, but the hardware will be developed first to accompany programs that purport to be lie detectors and others that help you generate alpha brain waves.

At this point, personal-computer game output is limited to what can be displayed on a computer monitor and piped to a speaker.

The motion-sensing and feedback cuffs I mentioned earlier will also debut as computer peripherals, but I don't see them as gaming devices right away. They'll find more serious (and expensive) acceptance in military training and medical applications.

An interesting wrinkle on some existing technologies will make possible true multiplayer games. Networking among individual computers—whether via modem or plug-in keyboards sharing one computer—will become popular. Games such as a multiterminal RPG proposed by science fiction writer John M. Ford are not more than five years away.

The intermediate future (1995-2010)

Give it a decade or so, and we'll have multipurpose, modular machines. (Yes, I know—Coleco tried it with the game machine that could be expanded into a computer and vice-versa. But that was a special case that failed because it was marketed as a computer to the dedicated game market, and it wasn't much of a computer anyway.) The modular machines will be thrown out in a variety of forms, but each will be clearly identifiable as a computer.

The range and sophistication of input devices will be staggering. Keyboards and joysticks and mice will still be with us, but the market will be exploding with new input devices, ranging from extremely sophisticated motion and eye-movement sensors (spin-offs from military, medical and space research) to workable voice-recognition systems. Waldos will be common, and overall, esoteric input devices will enjoy a renaissance of development.

Motion-cuing systems will be common, though still somewhat cumbersome. Many will take the form of chairs costing as much as computers themselves. Tactile feedback will still be in the market-and-fail stage, with the hardware required bulky and obtrusive—not to mention expensive—so don't look for it to be common until after the turn of the century.

All of these devices will first appear in arcades and amusement parks, but the cost will drop as the demand rises and technology enables lower cost production, which of course will make them practical as computer peripherals.

The long range (2010 and beyond)

It starts getting fuzzy at this point. There is always the potential for something like the microchip to come along. And there's always the problem of science fiction author Vernor Vinge's "singularity," which wonders how we can imagine what we cannot imagine. And nanotechnology presents a real wild card. As we move more than two decades into the future, the problem is not unlike that of Beaker trader trying to imagine television and telecommunications.

Besides which, as Arthur C. Clarke puts it, "Any sufficiently advanced technology is indistinguishable from magic." And if we could glimpse post-2010 computer technology, we might well see it as magic.

But, assuming we aren't confronted with totally incomprehensible developments, figure on being able to go out and buy a multisensory system that will take you anywhere (anywhere the marketing types figure a significant number of game buyers want to go, that is). The system will probably be sold for specialized machines as well as for the growing number of computer-based do-everything machines.

This system may well use crude brain wave input/output devices to monitor your responses and build your environment (see the accompanying sidebar). Whether or not direct mind links are possible, a typical gaming system in, say, the year 2020 will consist of a system that uses incredibly sophisticated versions of the input/output devices like I've already discussed. But you'll be no more aware of their presence while playing a game than you are of the disk in your computer's drive.

The exact nature of such games I won't even take a shot at; like fiction, gaming reflects society, and I'll leave it as an exercise for the reader to determine the makeup of society in 2020.

Postscript: games and society

It's worth spending a few words here to consider the effects of all this on society to be. The major effect is this: Gaming will be taken seriously by society at large. Look for computer games to become as common a form of entertainment as watching television. And the value of the simulation element of gaming will be recognized and assimilated by both the education and business community. (Just as it has already been accepted by the aerospace and military communities.)

In this respect, gaming will achieve its maturity.

ANALOG Computing *columnist Michael A. Banks is the author of several science fiction novels (among them* The Odysseus Solution, *from Baen Books) as well as numerous short stories. He's also written nonfiction books on telecommunications, rocketry and other topics. Banks' latest book,* The Modem Reference *(Brady Books, 1988), is a comprehensive guide to using modems and online services, now and in the future.*

A far-future gaming scenario

You slip a CD-ROM into your computer's game slot. The machine's status screen comes to life and a human voice issues from its speaker:

"Welcome to *The Arena.* If you are new to surrogate-gaming, please carefully read the instructions included in this game's package, as well as those accompanying your game-player.

"Signal when your machine's eyeoo device is in place. . . ."

The voice drones through the usual legal disclaimers and preliminary instructions—all of which you've heard 100 times before—as you adjust a plastic cap on your head. You cut the spiel short by telling the computer to begin the game. The last thing you hear as the world fades away is ". . .clap your hands three times to end the game. If you are killed or experience a medical emergency, the game will end automatically."

The arena is 100 feet on a side, surrounded by multitiered seating from which 5,000 battle fans roar their bloodlust. You see but a fraction of them from where you stand, beneath a stone portico at the so-called Portal of Life—the gladiators' entrance to the arena. Half of the 100 men sharing the small space with you are Secutors, wearing a heavy breastplate and half-armor, and carrying a small shield and gladius, the Roman short sword. The rest are armorless Retiarii, armed with weighted metal nets and barbed tridents. The game's randomizer has assigned you the role of Secutor.

Sweat pours from every inch of your body, and the heavy armor chafes your sides. After an interminable period of time, you join the others in ranks of four and move onto the sandy floor of the arena. Once there, each man seeks the most advantageous position for the coming battle. You move to a spot near a curve in the area wall. Before you can determine the whereabouts of potential enemies—Retiarii—a trumpet signals the beginning of combat.

A Retiarius's net flashes before you; up flies your shield in reflex action. The net falls away and its owner scrambles to retrieve it. You lunge forward to deliver a vicious stab, but your foe is faster. Net forgotten, he whirls to confront you. He lunges, jabs at you in what you realize too late is a feint, then jabs again. Three barbs sink deep into your right thigh. You groan in agony as you pull away, the trident still buried in your flesh.

The force of your retreat pulls the trident from your leg—and from the hands of its owner. You waste no time in taking advantage of this; adrenaline pumping, you plant a foot on the trident's shaft as the Retiarius scrambles for the weapon. He jerks, slips on loose sand and is down. A quick thrust with your sword dispatches him.

You adjust your breastplate and slog across the sand to help a fellow Secutor dispatch a netman, but something's wrong. Your leg grows heavy, all but refusing to follow your movements. You look down to see that you're loosing blood at an unbelievable rate. You pause, and another net finds you, an almost perfect cast from behind! You slash desperately at the net as you turn to face this new antagonist.

But the net is the least of your worries. The Retiarius is everywhere at once. The wicked prongs of his trident bite your flesh painfully time and again as you block and dodge. And your sword finds naught but empty air.

Then your leg gives out; you drop to the ground. You can barely see the Retiarius through the sweat and blood running in your eyes. A haze of pain overwhelms your consciousness. It is all you can do to hold on to your shield and sword.

You wave the gladius in feeble threat, and the crowd roars as the smiling victor raises his trident to deal the death blow. You drop the sword and raise your hand, palm upward, in the universal sign for mercy. The Retiarius pauses in mid-thrust, places one sandled foot on your chest, and looks to the crowd.

The decision is unanimous. Thousands of screaming fanatics jab their thumbs downward: death. Your conqueror's grin widens, and he raises the trident high in a two-handed grip. You struggle to move away, but you are too weak. With a cry of triumph, the Retiarius drives the trident deep into your midsection.

The agony is imaginary, brief and the transition from the game is almost instantaneous, but the memory remains. You lived that battle—and died. You unconsciously check your stomach and leg, just to make sure it was only a game.

Perhaps you're thinking, *There's no way that kind of gaming system will ever exist.* If so, imagine trying to describe *Ultima* or even *Zork* to the average citizen of 17th-Century Paris or Rome. Now, ask yourself if you can really say the preceding will *never* be possible.

But we're not all that close to electroencephlograph-like devices that can turn brain-wave patterns into computer input, *and* induce brain-wave patterns for player input. Science is a long way from understanding the human brain well enough to know in just what manner those patterns should be interpreted, and how to create patterns that accurately simulate direct sensory input to the brain. The capability to use EEG-type equipment to do more than induce random sensations is several decades away; ditto for reading minds with such equipment.

So, the gladiator game is more science fiction than science at this point. However, you never know when a breakthrough will occur. Look at what the transistor and the microchip did to our world.

Just for fun

If you find speculation on man-machine interfaces interesting, I highly recommend these science fiction novels, currently available in paperback:

Mind Players, by Pat Cadigan,
Neuromancer, by William Gibson,
The Peace War, by Vernor Vinge,
True Names and Other Dangers, by Vernor Vinge.

—MAB

Assembly Line

NUMBERS, PART I

by Douglas Weir

```
DC.W %1100000000000011 ;#4
DC.W %0000000000000000 ;#5 BK
DC.W %1100000000000011 ;#5 FG
DC.W %0000000000000000 ;#6 BK
DC.W %0000011111100000 ;#6 FG
DC.W %0000000000000000 ;#7 BK
DC.W %0000011111100000 ;#7 FG
DC.W %0000000000000000 ;#8 BK
DC.W %0000011111100000 ;#8 FG
DC.W %0000000000000000 ;#9 BK
DC.W %0000011111100000 ;#9 FG
DC.W %0000000000000000 ;#10 BK
DC.W %1100000000000011 ;#10 FG
DC.W %0000000000000000 ;#11 BK
DC.W %1100000000000011 ;#11 FG
DC.W %0000000000000000 ;#12 BK
DC.W %1100000000000011 ;#12 FG
DC.W %0000000000000000 ;#13 BK
DC.W %1100001111000011 ;#13 FG
DC.W %0000001111000011 ;#14 BK
DC.W %1111111111111111 ;#14 FG
DC.W %0000000000000000 ;#15 BK
DC.W %1111111111111111 ;#15 FG
```

Illustration by John Berado

I think that now is a good time to start talking about numbers. As we've seen, the 68000 at the lowest level works only with numbers; so it's more than a little ironic that the assembly-language programmer usually has to go to much more trouble to write even simple number-crunching programs than he or she would for string manipulation programs.

Much of the problem, of course, lies in handling the I/O. The "numbers" typed at the keyboard have to be translated from character strings into integral binary values that can be added, subtracted, multiplied, or whatever; and when it comes time to output the results of these calculations, the numbers have to be translated back into character strings. There's a lot more to this than meets the eye, and it's one reason why routines like *print()* in C are large and complicated. This month we'll take a look at converting unsigned integers back and forth, and in the process learn something about multiplication in the 68000.

This is something of a chicken-and-egg situation. We could write math routines to our hearts' content, but we'd have no idea how accurate the routines were unless we could look at (i.e., output) the results. But it turns out that the routines required for input and output require a fair amount of computation themselves, so writing a set of conversion routines is a good first step in more ways than one. This month's program does nothing more than repeatedly input two integer strings from the keyboard, convert them into binary values, multiply them, convert the result into a string, and print it. This, however, requires a lot of multiplication and division. Let's see how it's done.

The awful truth about multiplication

Let's think 32 bits. We know that the 68000's data registers are 32 bits long, so it seems reasonable to assume that we can multiply numbers up to 32 bits long with the 68000's multiply instruction.

Unfortunately, this is not a correct assumption. The 68000 does indeed have a multiplication instruction, but it can handle operands only up to 16 bits long. This is because multiplying two numbers almost always produces a result with more digits than either of the operands; in fact, the operand can be as much as twice as long as two equal-length operands. This is just as true of decimal numbers as it is of binary; for example, 99 * 99 = 9801. Since the result of the multiply instruction is stored in a single-data register, it's clear that neither operand can be greater than 16 bits. In other words, if you are multiplying unsigned integers, the largest operand you can have is 65535.

Note that it's perfectly possible to have one operand much longer than 16 bits and still come out with a product that could be held in 32 bits, as long as the other operand was small enough. For example, 99999 * 1 = 99999, which is only 17 bits long. Nevertheless, you can't do this example on the 68000 without writing some routines of your own, because of the arbitrary restriction of 16-bit operands. This month we will take the first step toward writing a general multiple-precision multiplication routine. For now, we want to have the ability to multiply any pair of integral numbers, no matter what their sizes, so long as their product doesn't exceed the 32-bit limit.

Now let's take a look at the 68000's *mulu* (multiply unsigned) instruction, which will be the basis of our routine. In order to multiply two unsigned 16-bit (or smaller) integers using this instruction, you load each operand into the low word of a data register, and then specify these two registers as the source and destination of the instruction. For example:

```
        move.w      #1685,d0
        move.w      #1756,d1
        mulu        d0,d1
```

In these three lines of code, the integers 1685 and 1756 are multiplied. The 68000 puts the product of the multiplication into the destination register. In other words, the original contents of the destination register are always lost when a *mulu* is executed. Since the product can be up to 32 bits long, all 32 bits of the destination register are written to by *mulu*.

The ideal multiply

What we would like to do is multiply the entire contents of two 32-bit registers, thus obtaining a 64-bit product. For the time being, we'll ignore the "high" 32 bits of such a product, and just trust ourselves to choose only those pairs of operands whose products won't exceed 32 bits in length. This is analogous to multiplying decimal numbers with, say, two decimal places each instead of one decimal place.

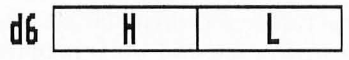
With decimal numbers, multiplying single-place integers is easy, assuming you know your multiplication tables. For example, 9 * 9 = 81, 4 * 6 = 24, and so on. To multiply a couple of two-digit numbers, we usually need a pencil and paper:

```
   24
 * 48
 -----
    32  (8 * 4)
   16   (8 * 2)
   16   (4 * 4)
   8    (4 * 2)
 -----
 1152
```

Here, a series of partial products is added to get a final product. Doing the partial products lets us multiply larger numbers as if they were made up of smaller ones; we sort of ignore the extra decimal places, but everything comes out all right in the end because of the way the partial products are grouped and added. Thus, the partial product 8 * 2 is written beginning in the 10's column, since the 2 is actually a 20; and so on.

We can use this same technique to multiply the contents of the 68000's binary data registers. Suppose we wanted to multiply the contents of registers d5 and d6. Without bothering about the actual configuration of bits in the registers, we can imagine the numbers in them being arranged as in Figure 1, where each of the two registers is made up of a "high" word (16 bits) and a "low" word, represented by H and L, respectively.

FIGURE 1



What we need to do

In a normal 68000 *mulu* operation, only the two L portions of the registers would be multiplied, much as the rightmost digits of 24 and 48 were multiplied in the decimal example above. If we want to write a routine to multiply all 32 bits of both registers, then we have to do much the same thing as we did with the two-digit decimal numbers. First we must multiply d5's low word with d6's low word; then d6's low word with d5's high word (and we must remember to add this partial product in at the right binary digit place, just as we did with the decimal product); then d6's high word with d5's low word; and finally, the two high words of d5 and d6. The sum of all these partial products, correctly done, will be the full 64-bit product of the two 32-bit registers. The arrangement of the partial products for the final sum would look something like Figure 2.

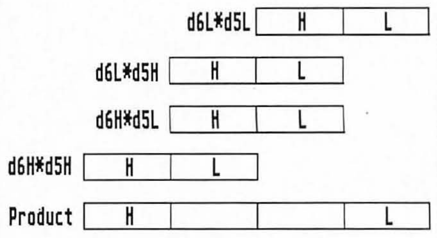


FIGURE 2

In other words, suppose we've reserved eight bytes of memory for the product of the two registers. We now multiply the low words of the two registers, and store the result in the low longword of our storage area. Next we multiply the low word of d6 with the high word of d5. We add this result to the "middle" longword of our storage area. Likewise with the next partial product (from the high word of d6 and the low word of d5). Finally, we add the last partial product (from the two high words of the registers) to the high longword of the storage area.

This month's program

And that's just what the routine *mul32* in this month's program does. It receives its two 32-bit operands on the stack, multiplies them, and replaces the operands with the 64-bit product, so that the caller can simply pop it off the stack after the subroutine returns.

Let's look at *mul32* in more detail. After clearing several data registers, I load register a1 with the base address of a table of addresses of three tiny subroutines—I'll explain what these do a little later on. I load the two operands from the stack into data registers d5 and d6, and load the address of the "top" operand (i.e., the first pushed, and thus the higher in memory) into register a0. To do this, I used a new instruction, *lea* (load effective address).

We discussed the concept of "effective address" several installments ago. In essence, the effective address of an operand can be said to be "where the processor goes to get the value for the operand." The *lea* instruction simply calculates the address for the operand in the source field, and loads this address (not the operand) into the destination operand. Here, the effect of *lea* is that a0 is loaded with the address of the first of the two operands pushed. The next two instructions clear the 64-bit area originally occupied by the operands; this is where the product will be returned.

It's actually quite easy to get at the high and low words of the two operands in order to calculate the partial products. The 68000 *swap* instruction exchanges the 16-bit halves of any data register, making the high word the low, and vice versa. Once we have the correct word in the bottom half of the register, it can be multiplied with *mulu*. However, since *mulu* always destroys the original contents of its destination register, we need to move the current word-operands into a set of "temporary" registers for the actual computation.

This is done at the top of *m_loop*. The low words of d5 and d6 are moved into d2 and d3 and multiplied. Now comes some special checking. Remember, we're adding these partial products into a cumulative total. That means we have to keep in mind the possibility of a carry from the previous addition. This occurs in much the same circumstances that it would in decimal addition. If, for example, we were to add 19 + 19, then we would first add the two 9s, get 18, and write the 8 in the 1's column. The 1 would be carried over into the 10's column and added in with the two 1s there.

Carrying on in binary

When, for instance, two 1s are added in binary arithmetic, the following happens:

```
  1  ( 1 decimal)
 +1  (+1 decimal)
----
 10  ( 2 decimal)
```

Here, a zero is entered in the current place, and the one is carried over one place. Obviously, this sort of carry occurs quite often in binary arithmetic, and the process of carrying the ones over is completely transparent to the programmer—unless the bit from which the one is carried happens to be the leftmost bit of the data size of the operand. In other words, imagine what happens if you add the low word contents of two registers and there is a carry out of the 16th (i.e., leftmost, since this is a word) bit of the result. In this case, the bit has nowhere to go, and the programmer has to be warned.

This warning is accomplished by setting the Carry flag in the Condition Codes register. Whenever we add a partial product to our cumulative sum, the Carry flag will be set if there has been a carry out of the leftmost bit of the destination. So we have an easy way of finding out about carries. However, it's not so convenient to do anything about it at the moment it's detected. Why not? Because the carry will have to be added into the next part of the sum, and we haven't gotten to it yet. So we have to save the condition, since, as we know, the Condition Codes change constantly, almost with every instruction. Fortunately, there's a very simple way to do this.

If you look down a few lines to the next label, you'll see (under the add) the instruction *scs*. This is just one form of a versatile instruction that allows you to save any condition in a specified destination. Here scs means "set if carry set." If the Carry flag is set when this instruction is executed, then the destination (always byte-sized) will be loaded with all 1s; i.e., all bits of the destination byte will be set if the Carry flag is currently set. If the Carry flag is not set, the destination will be cleared (loaded with 0s). This instruction is listed as *scc* (set according to condition) in the manuals, similar to *bcc* (the branch instructions). There you can find the correct form of the instruction to save whatever condition you want.

Now let's return to the instruction after *mulu*. The instruction *tst* simply "tests" the contents of its operand and sets the Condition Codes accordingly. After testing d4, we can find out if a previous carry has been saved in it or not. If d4 contains zero, then there was no carry, and execution proceeds to the label, *m_nxt0*; otherwise, 1 is added to the current partial product which will be added to the cumulative sum.

Adding the products

In fact, this addition is the next thing that is done. The double longword into which we are adding the products is indexed with the register d0. The first time through this loop, a0 is pointing to the low longword and d0 contains zero, so the first partial product goes into the low longword of the product. The next couple of times, we want to add into the "middle" longword of the product. This is done by subtracting two from d0. With the indexed addressing, this is the same as subtracting two from the address in a0 and thus pointing one word left or "higher up" in the product. The last partial product is added into the high longword, and we point to it by again subtracting two from d0.

How do we know when to subtract from d0? We don't want to do it on every iteration—the second and third partial products have to be added into the same longword area. Since it turns out that we want to subtract on odd iterations (the first and third) only, all we have to do is find out if we're on an odd iteration, and if so, subtract. This is done with our next-to-last new instruction, *btst*.

Computer Garden

Wilkes-Barre & Scranton's Favorite Computer Store

Accolade
Hardball............$25
Pinball Wizard...$23
Test Drive.........$25
Antic
Spectrum 512 ...$56
Cyberstudio......$70
Atari
520STFM........ $Call
1040ST........... $Call
Mega-2 ST..... $Call
Mega-4 ST..... $Call
Avant-Garde
PC-Ditto......... $Call
Avatex
1200e modem...$79
2400 modem ...$169
Modem cable....$15
Batt.Included
DEGAS.............$49
Digital Vision
Computereyes$179
Dr. T's
KCS.................$189
The Copyist $189
MIDI Recording Studio $26
Epyx
Impos. Mission II$25
FTL
Dungeonmaster$25
Sundog............$25
Future Sys.
GTS-100 drive $199
ICD
F20A-ST drive $669

Intersect
Interlink $25
LC Tech.
Stereotek kit..$139
Megamax
Laser C..........$159
Michtron
GFA Basic.........$39
GFA Compiler....$39
GFA Companion$33
GFA Basic book$25
Microprose
F-15 Strike Egl..$25
Gunship............$33
Silent Service...$25
Microsoft
Microsoft Write $89
Panasonic
1080i-II Printer.$175
1091i-II Printer.$195
Printer cover $9
Printer cable.....$15
Practical Per.
Monitor Master.$44
Mouse Master ..$35
SeymorRadix
IMG-Scan..........$79
Sierra
Space Quest $33
Space Quest II..$33
King's Quest $33
King's Quest II...$33
King's Quest III..$33
Donald Duck.....$16
Police Quest.....$33
Liesure Suit $25
Mother Goose ..$20

Soft Logik
Pub. Partner $60
Pub. Prtnr Pro.$120
Font Disk 1or 2.$20
Springboard
Certificate Mkr.$33
SSI
Wizard's Crown $25
Rings of Zilfin ...$25
Phantasie..........$25
Phantasie II.......$25
Phantasie III $25
War Game Con.$23
Star
NX1000 Printer$169
Color version..$229
Printer cable.....$15
Sublogic
Flight Simulator.$33
Scenery Disk 7.$15
Scenery Dsk 11 $15
Supra
2400 modem ...$139
Modem cable....$15
Timeworks
Wordwriter $49
Datamanager ...$49
Swiftcalc $49
Partner $33
Desktop Publish$79
Trio Eng.
Digispec............$39
Unispec.............$49
Unison World
Printmaster + ...$25
FantasyArtGlry $20
Fonts&Borders $23

Toll-free order line:
1-800-456-5689
For information call 1-717-823-4025
3% charge for VISA-MC-AMEX. Shipping, extra.
Computer Garden, 106 W. Carey St. Plains PA 18705

CIRCLE #119 ON READER SERVICE CARD.

The *btst* instruction allows you to test the state of any single bit in the destination operand. The number of the bit (counting from 0 as the rightmost bit) can be specified as an immediate value, as it is here, or it can be contained in a register. If the bit is set, then the Zero flag in the Condition Codes register will be cleared. If the bit is not set, then the Zero flag will be set.

After counting the loop in d1, I test d1's rightmost bit. From our binary arithmetic, we know that each bit in a register stands for a power of two, counting from the rightmost bit. Bit 0, then, (the bit we are testing), stands for the value of two to the zero power, i.e., 1. This is the only power of two with an odd value, so if this bit is set, we know that the register as a whole contains an odd value. So, after doing the *btst*, the Zero flag will be cleared if d1's rightmost bit is set, the *beq* will not be executed, and the immediate value 2 will be subtracted from d0. If the bit isn't set, then the branch will be taken, and the subtract from d0 will be skipped this time.

The indirect jump

Finally, we have to do some swapping of d5 and d6 on every iteration in order to get the desired partial products. This is handled with the trio of tiny subroutines *m__swap__1*, *m__swap__2* and *m__swap__3*. Remember that we loaded the base address of a table which contains the addresses of these routines into a1 at the beginning of the routine. Since each address in the table is four bytes long, we can index through this array of addresses by adding offsets of 4 to the value of a1. Each time through the loop, the next subroutine address is indexed and loaded into a2.

The address now contained in a2 is essentially the same thing as the label that identifies a subroutine in the destination field of a *bsr* instruction: The label is, as we know, really an address. We can "jump" to the address contained in any address register by executing the *jsr* (jump to subroutine) instruction. You can't use the *bsr* instruction here—I'll explain why next time.

Thus, on every iteration of the loop, we index to the address of a subroutine that contains the correct swap instructions to set up the data registers for the next *mulu*, and we then execute that subroutine by "jumping" to the value in a2 "indirect." (We're jumping to the address contained in a2—hence the parentheses—not to a2 itself!) As part of its operation, *jsr* pushes the return address onto the stack, just as *bsr* does. Thus a simple *rts* at the end of each subroutine returns us to the next instruction after the *jsr*.

And that's just about all there is to it. When *mul32* terminates, its caller can pop the high and low longwords of the product off the stack. The caller would have to adjust the stack anyway, since that's where the operands were passed in the first place, so not much time and no space is wasted.

Next time

We'll look at the rest of this program in detail next month. In the meantime, type in the source code and assemble it. When run, the program will wait for you to type in two numbers, each followed by a carriage return. After you've entered the second number, the program will multiply them and display the answer. You can repeat this process as often as you like—the program won't terminate until you press CONTROL-C. If you type a number that's too large for the program to handle, or if the product of two numbers is too large, the program will display an error message (the message is the same in all cases), and terminate.

And now I have a small confession to make. The version of *mul32* given here is probably not the simplest way to do 32-bit multiplication, but I took advantage of the opportunity to introduce some little-used instructions. You can try your hand at other methods—just rewrite the subroutine to your taste, and as long as it receives its operands and returns its answer in the same way as the original version, it should work fine with the other routines (don't forget to save and restore at the beginning and end of your routine all the registers you use). As for the approach you should take, here's a hint: Try not using a loop.

Assembly Line
Listing 1-Assembly

```
****************************************************************************
*
* Assembly Line Nr 8: Multiplying and Converting Numbers
*
* by Douglas Weir
*
* Copyright 1988, ST-Log
*
****************************************************************************

        text

getnbr:
        move.l  #str_top,-(sp)          push address of template
        move.w  #$0a,-(sp)              code=read edited string
        trap    #1                      do it
        addq.l  #6,sp                   pop args
        movea.l #string,a0              get address of string
        clr.l   d0                      clear d0 for indexing
        move.b  str_cnt,d0              get nr of chars input
        move.b  #0,0(a0,d0.l)           insert a null terminator
        move.l  #string,-(sp)           get address of string
        bsr     cvnbr                   convert to number
        addq.l  #4,sp                   pop arg
        move.l  d0,d7                   save conversion
        bsr     c_return                print newline

        move.l  #str_top,-(sp)          push address of template
        move.w  #$0a,-(sp)              code=read edited string
        trap    #1                      do it
        addq.l  #6,sp                   pop args
        movea.l #string,a0              get address of string
        clr.l   d0                      clear d0 for indexing
        move.b  str_cnt,d0              get nr of chars input
        move.b  #0,0(a0,d0.l)           insert a null terminator
        move.l  #string,-(sp)           get address of string
        bsr     cvnbr                   convert to number
        addq.l  #4,sp                   pop arg
        bsr     c_return                print newline

        cmp.l   #MAXOP,d7               is first operand too big?
        bpl     over32                  if so, quit
        cmp.l   #MAXOP,d0               is second operand too big?
        bpl     over32                  if so, quit
        move.l  d7,-(sp)                get first operand
        move.l  d0,-(sp)                get second operand

        bsr     mul32                   multiply them
        move.l  (sp)+,d0                discard high longword
        bne     over32                  if overflow

        move.l  (sp)+,d0                get low longword
        move.w  #FALSE,-(sp)            FALSE==don't print leading zeros
        bsr     cv32                    convert to ascii
        addq.l  #2,sp                   pop arg

        move.l  #dgts,-(sp)             get address of string
        move.w  #$09,-(sp)              code=print string
        trap    #1                      do it
        addq.l  #6,sp                   pop args

        bsr     c_return                print a newline
        bra     getnbr                  go back to start...

over32:
        move.l  #oo_string,-(sp)        "input overflow" string
        move.w  #$09,-(sp)              code=print string
        trap    #1                      do it

        move.w  #0,-(sp)                code=terminate
        trap    #1                      do it

*****
*
* c_return-- prints a newline to the screen.
*
*****
c_return:
        movem.l d0-d2/a0-a2,-(sp)       save registers
        move.w  #$0d,-(sp)              carriage return
        move.w  #2,-(sp)                code=print character
        trap    #1                      do it
        addq.l  #4,sp                   pop args
        move.w  #$0a,-(sp)              line feed
        move.w  #2,-(sp)                code=print character
        trap    #1                      do it
        addq.l  #4,sp                   pop args
        movem.l (sp)+,d0-d2/a0-a2       restore registers
        rts                             and return
```

PROGRAMMING

```
*****
*
* div32-- performs stupid 32-bit division.
*
* at entry:
*               d0.l contains dividend
*               d3.l contains divisor
* at exit:
*               d0.l contains quotient
*               d1.l contains remainder
*               all other registers preserved
*
*****
div32:
        movem.l d2-d7,-(sp)             save registers
        clr.l   d4                      clear subtract counter
        tst.l   d3                      divide by zero?
        beq.s   div_zero                if so
div_loop:
        cmp.l   d3,d0                   ok to subtract?
        bcc.s   div_count               if so
        bra.s   div_out                 else we're done
div_count:
        sub.l   d3,d0                   do subtract
        addq.l  #1,d4                   count it
        bra.s   div_loop                and continue
div_out:
        move.l  d0,d1                   return remainder
        move.l  d4,d0                   return quotient
div_exit:
        movem.l (sp)+,d2-d7             restore registers
        rts                             and return

div_zero:
        move.l  d0,d1
        move.l  #BAD,d0                 error code
        bra.s   div_exit                and leave


*****
*
* cv32-- converts an unsigned 32-bit binary number to ascii.
*       The largest possible unsigned 32-bit number is 4294967295.
*       This program sets a maximum of MAXOP (see below).
* at entry:
*               d0.l contains number to be converted.
*               (a6 + 8) -> leading zeros flag (word):
*                           TRUE => do print leading zeros
*                           FALSE => do not print leading zeros
* at exit:
*               'dgts' (in data segment) contains converted number.
*               all registers preserved.
*
*****
cv32:
        link    a6,#0                   link register
        movem.l d0-d4/a0-a1,-(sp)       save registers
        movea.l #dgts_end,a0            point to end of string space
        move.b  #0,(a0)                 insert null terminator
        moveq.l #d_len,d1               get length of string space
        bra.s   cv_test                 clear string space
cv_iloop:
        move.b  #BLANK,-(a0)            clear a byte
cv_test:
        dbra    d1,cv_iloop             go till end
        tst.w   8(a6)                   check leading zero flag
        beq.s   cv_nxt0                 if FALSE
        move.l  #FALSE,d4               otherwise, don't check
        bra.s   cv_nxt1
cv_nxt0:
        move.l  #TRUE,d4                leading zeros = true
cv_nxt1:
        movea.l #divisors,a1            point to start of array
        move.l  (a1)+,d3                get first divisor
        moveq.l #d_len,d2               get place index
        neg.l   d2                      index from back to front
        movea.l #dgts_end,a0            point to end of string space
cv_loop:
        bsr     div32                   do 32-bit division

        cmp.l   #TRUE,d4                leading zeros?
        bne.s   cv_nxt2                 if not
        tst.b   d0                      else: is this a zero?
        beq.s   cv_nxt3                 if so
        move.l  #FALSE,d4               else set flag = not leading zeros
        bra.s   cv_nxt2                 and go insert digit
cv_nxt3:
        move.b  #BLANK,d0               use space vice leading zero
        bra.s   cv_nxt4                 and insert it
cv_nxt2:
        addi.b  #ZERO,d0                convert to ascii
cv_nxt4:
        move.b  d0,0(a0,d2.l)           insert into string space
        move.l  d1,d0                   recover remainder
        move.l  (a1)+,d3                get next divisor
        addq.l  #1,d2                   decrement place index
        bne.s   cv_loop                 go till end

        cmp.l   #TRUE,d4                was number a zero?
        bne.s   cv_exit                 if not
        subq.l  #1,d2                   else find last digit
        move.b  #ZERO,0(a0,d2.l)        and insert a zero
cv_exit:
        movem.l (sp)+,d0-d4/a0-a1       restore registers
        unlk    a6                      deallocate frame
        rts                             and return


*****
*
* cvnbr-- converts a string of ascii digits to a number.
*         since the high longword from 'mul32' is at present discarded,
*               only a conversion of the low 32 bits is returned.
*         the digits themselves are not checked for errors.
* at entry:
*               (a6 + 8) -> string to convert (null-terminated).
* at exit:
*               d0 contains converted number, -1 (BAD) if error.
*               all other registers preserved.
*
*****
cvnbr:
        link    a6,#0                   frame pointer
        movem.l d1-d2/a0-a1,-(sp)       save registers
        movea.l 8(a6),a0                get string
        clr.l   d0                      clear counter
        move.w  #MAX_DIGITS+1,d1        max size of string
        bra     front_test              get least sig digit
front_loop:
        addq.l  #1,d0                   count one place
        tst.b   (a0)+                   null?
front_test:
        dbeq    d1,front_loop           if not and <= max length

        tst.w   d1                      too long?
        bmi     front_error             if so
        subq.l  #1,d0                   else uncount delimiter
        subq.l  #1,a0                   point back to null
        clr.l   d1                      clear digit holder
        clr.l   d2                      clear number holder
        movea.l #factors,a1             point to array of factors
        bra     cvn_test                start conversion
cvn_loop:
        clr.l   d1                      clear upper bytes
        move.b  -(a0),d1                get a digit
        subi.b  #ZERO,d1                convert it from ascii
        move.l  d1,-(sp)                push as first operand
        move.l  -(a1),-(sp)             current factor = 2nd operand
        bsr     mul32                   multiply them
        move.l  (sp)+,d1                discard high longword
        move.l  (sp)+,d1                low longword = product
        add.l   d1,d2                   add to total
cvn_test:
        dbra    d0,cvn_loop             continue to end
cvn_exit:
        move.l  d2,d0                   return number
        movem.l (sp)+,d1-d2/a0-a1       restore registers
        unlk    a6                      deallocate frame
        rts
front_error:
        move.l  #BAD,d2                 error code
        bra     cvn_exit                now leave


*****
*
* mul32-- performs complicated 32-bit multiplication.
*
* at entry:
*               (a6 + 8)  -> first operand.
*               (a6 + 12) -> second operand.
* at exit:
*               after return to caller,
*               (a7) -> high longword of product.
*               (a7 + 4) -> low longword of product.
*
* all registers preserved.
*
*****
mul32:
        link    a6,#0                   frame pointer
        movem.l d0-d7/a0-a2,-(sp)       save registers

        clr.l   d0                      clear registers
        clr.l   d1
        clr.l   d4
        clr.l   d7
        movea.l #m_subs,a1              base address of swap routines
        move.l  8(a6),d5                get operands
        move.l  12(a6),d6
        lea     12(a6),a0               point to result area, low word
        clr.l   8(a6)                   clear product area
        clr.l   12(a6)
m_loop:
        move.w  d5,d2                   get partial operands
        move.w  d6,d3
        mulu    d2,d3                   multiply them

        tst.b   d4                      carry from last add?
        beq.s   m_nxt0                  if not
        addq.l  #1,d3                   else add it in
m_nxt0:
        add.l   d3,0(a0,d0.l)           add partial product
        scs     d4                      save possible carry
        addq.l  #1,d1                   count loop
        btst.l  #0,d1                   odd iteration?
        beq     m_nxt1                  if not
        subq.l  #2,d0                   else index product up one word
m_nxt1:
        cmp.l   #12,d7                  finished?
        beq.s   m_out                   if so
        movea.l 0(a1,d7.l),a2           else get swap routine
        jsr     (a2)                    execute it
        addq.l  #4,d7                   index for next one
        bra     m_loop                  and continue
m_out:
        movem.l (sp)+,d0-d7/a0-a2       restore registers
        unlk    a6                      deallocate frame
        rts                             and return

m_swap_1:
        swap    d5
        rts
m_swap_2:
        swap    d5
        swap    d6
        rts
m_swap_3:
        swap    d5
        rts
```

```
*****
*
* data segment
*
*****
                data
                even
m_subs          equ     *       address table of subs for 'mul32'
                dc.l    m_swap_1
                dc.l    m_swap_2
                dc.l    m_swap_3

MAX_DIGITS      equ     10      max length of digit string
BAD             equ     -1      error code
ZERO            equ     48      ascii '0'
FALSE           equ     0       false constant
TRUE            equ     1       true constant
BLANK           equ     32      ascii ' '
MAXOP           equ     400000000       max size (+1) of operand allowed

oo_string       dc.b    'oops... answer is too big',$0d,$0a,0
dgts            ds.b    10      string area for ascii number
d_len           equ     10      length of string space
dgts_end        dc.b    0       null at end of string

                even
divisors        dc.l    1000000000      array of divisors and factors
                dc.l    100000000
                dc.l    10000000
                dc.l    1000000
                dc.l    100000
                dc.l    10000
                dc.l    1000
                dc.l    100
                dc.l    10
                dc.l    1
factors         dc.l    0

str_top         dc.b    81      max nr of chars to input
str_cnt         ds.b    1       byte for count
string          ds.b    80      beginning of actual string
```

for ATARI ST's and MEGA's

MEMORY upgrades:

Solderless "plug in" installation, 1 year warranty
520ST – expand to 1, 2.5 or 4 MB on ONE board – prices start at $129 for the 0K version – or go to
1 Megabyte only, socketed, 0 K $ 79
1040/520STfm – upgrade to 2.5 and 4 MB, fully socketed 0 K board $149.
For all our memory upgrades: on board CLOCK module only $30 including software!

For more detailed catalog contact:

tech-specialities Co.
1022 Hodgkins, Houston, TX 77032
☎ (713) 590-2068 and 590-3738

We ship COD or prepaid, sorry, no credit cards!
Shipping/Handling on memory upgrades $ 5
on Hard Drive Kits – $10/no drive – $20/w. dr.
Texas residents add 8% state sales tax.

EXPANDABLE Hard Drive Kits:

1. 10" x 6.8" x 15", full SCSI interface with DMA through – 150 W PC power supply with fan – room for up to 5 half ht. drives – mounts on floor, under desk or on desktop – can supply power to 520ST and disk drives with optional cable set.
with 40 MB full height 30 ms drive $745
No Drive...install your own $385
2. MEGA footprint, 3.8" high, full SCSI/DMA-through interface, room and power for 3 half height or 1 each full/half height 5 1/4" drive, with fan.
with 40 MB half height 3.5" – 47 ms drive $695
with 10 MB 5-1/4" drive $395
3. 4.5" wide x 6" high x 13" deep, full SCSI/DMA-through – ready for 2 half ht. or 1 full height.
with 20 MB 1/2 height $485
No Drive...install your own $249

Atari 520ST, 1040St, 520STfm and MEGA are trademarks of ATARI Corp.

CIRCLE #120 ON READER SERVICE CARD.

C-

Last time we looked at a short utility that will set the ST's date and time. Though that program works fine when run from the Desktop, it would be more convenient to have it as a desk accessory, so that it would be available to us from within other GEM programs. Programming a desk accessory isn't much more complicated than programming any other GEM application, but, in the case of our Date/Time utility, there arises one complication.

How often have you seen a desk accessory that requires a .RSC file? Not too often. Oh, there's a couple of them floating around, but it is not a good programming practice to force a desk accessory to rely on a .RSC file—and for a very good reason.

A desk accessory, once it has been loaded, is constantly active. Even if you haven't selected it from the Desk drop-down menu, it is still running, waiting for the cue that will set it in motion. In fact, the only way to terminate a desk accessory is to shut off your machine.

Now think about what we know about .RSC files. They have to be loaded into the ST's memory with a call to *rsrc_load()*, but more importantly, the memory the resource file takes up has to be returned to the system at the termination of a program by a call to *rsrc_free()*. Since a desk accessory is always "running," when do we return the memory used by our resource trees?

I know what you're thinking. "What difference does it make whether or not we ever call *rsrc_free* when the only way to terminate a desk accessory is to turn the machine off? I mean, last time I heard turning off the machine was a *great* way to release all the memory!" Right you are. But there is one situation where a desk accessory gets reinitialized: when you switch resolutions. If your desk accessory has to load a resource file, then each time you switch resolutions, more memory will be taken up, because the resource is being reloaded, even though the old one hasn't been released.

So before we can convert our Date/Time utility to a desk accessory, we have to build our resource tree right into the program. How complicated this process will be depends on what tools you have available. If you have a resource construction program that'll save your object trees out in source code form, then you're three-quarters of the way there. All you have to do is plug in a few addresses and you're on your way. If you don't have access to a RCP that will do this for you, you'll have to write your resource by hand, a tedious project indeed.

In either case, though, we're going to have to have a clear understanding of resource trees in order to get our desk accessory's dialog box working. This means we'll be doing a review of some material and applying what we learn directly to our Date/Time utility.

Our resource tree

Take a look at Listing 1. Near the top you'll see some data labeled "Resource tree." This is all the data for our dialog box as it was saved from Atari's RCS2 in source code form. (Actually, RCS2 isn't too bright and saves data for *everything*, even data structures not used in our tree. I've already deleted the unneeded data for the sake of clarity.)

At the top of the data, you'll see an array of pointers called *rs_strings[]*. (Remember, even though the strings themselves are shown in this array, we still have an array of pointers here, each pointer holding the address of its associated string.) These pointers point to all the strings we need for our dialog box, with some of the objects being allotted three strings. (The first nine strings shown are three groups of three.) Why three? Because editable text fields require not only a text string but a format template and validation string as well. Remember?

The first string shown in *rs_strings[]* is our dialog-box's title line. We don't want an editable text field here so the format and validation strings (the second and third strings shown in *rs_strings[]*) are empty. The next group of three strings is for the time field of our dialog box. First is the *te_ptext* string (the text that will be displayed when the dialog box is first drawn and the area where the text the user enters will be found after he exits the dialog box), followed by the *te_ptmplt* string (the uneditable text that is displayed in the text field) and the *te_pvalid* string (the string that determines what type of data is allowable in each position of the string). If all of this is confusing to you, please review the "C-manship" articles on dialog-boxes that appeared in Issues 13 and 14 of ST-Log.

The next three strings are the *te_ptext*, *te_ptmplt* and *te_pvalid* strings for our dialog-box's date field. And finally there is the text for our OK and CANCEL buttons.

Following *rs_strings[]* is another array called *rs_tedinfo[]*. As you can tell by the name of the array, the data here makes up the tedinfo structures for the editable text strings in our dialog box. If you think back, you'll remember that a tedinfo structure contains all the information GEM needs to draw and handle editable text fields. There are three tedinfos contained in *rs_tedinfo[]*, one each for our dialog's title, time and date fields.

The first three long words of a tedinfo structure are pointers to the object's *te_ptext*, *te_ptmplt* and *te_pvalid* strings, respectively. If you look at the first three long words in the first tedinfo structure shown in *rs_tedinfo[]*, you'll see that we've got the values 0L, 1L and 2L. These don't look very much like pointers, do they? That's because they're not! They are actually offsets into the *rs_strings[]* array, telling us which pointers we need to place in the tedinfo structure.

This tedinfo is for our dialog's title field. We are told here that the *te_ptext* string for this field is pointed to by element 0 of the *rs_strings[]* array, and the *te_ptmplt* and *te_pvalid* strings for this field are pointed to by elements 1 and 2, respectively. When we initialize our desk accessory, it is up to us to see that the right

-MANSHIP

DESK ACCESSORIES AND BUILT-IN RESOURCE TREES

BY CLAYTON WALNUM

pointers get placed into the tedinfo.

The next six members of the tedinfo structure contain (in order) the font size, a reserved word, the horizontal justification of the text, color information, another reserved word and the border thickness for the object. These fields were all filled in by the resource construction program, so unless we are building our dialog box "by hand," we don't need to pay much attention to this data.

The last two members of the tedinfo are the length of the text string (*te_ptext*) and the length of the template string (*te_ptmplt*), and were also filled in by the resource construction program. You can see here that the length of our text is shown as 14. If we count the characters in our title, "DATE AND TIME," we'll see that we have 13 characters. Add one for the required null, and we've got 14. The template string contains nothing but a null, so it has a length of 1.

The next two tedinfo structures in the *rs_tedinfo[ ]* array contain the information for our time and date fields. The first three elements of each of the tedinfos are the only ones we are really concerned with. It is here that we have to place the pointers to the *te_ptext*, *te_ptmplt* and *te_pvalid* strings.

Moving down to the next array in Listing 1, we get to *rs_object[ ]* which is our actual object tree, made up of six different objects. The first object is the box that will hold all of the parts of our dialog, the second object is our dialog's title field, the third and fourth objects are our time and date fields, and the fifth and sixth objects are the OK and CANCEL buttons.

The first three members of an object structure are the numbers of the next sibling, the first child and the last child. The next member is the object type. In the first object shown in our listing, the object type is G_BOX, which is defined as 20 in your obdefs.h file. Next are the *ob_flags* and *ob_state* fields which tell GEM how to draw the object and what its current state is. We don't have to be too concerned with the information in these six members, as the necessary values were filled in by the resource construction program.

The seventh member of the object structure, however, is very important to us as we set up our "code resident" dialog box. It is the *ob_spec* field which, in the case of editable text objects, is where we must put a pointer to the appropriate tedinfo structure. If you look at this field in the first object, you'll see the value 0x21121L. This value contains information on the box's color and border thickness. It doesn't hold a pointer to a tedinfo because a G_BOX contains no editable text.

But look at the second object in the tree. This is our dialog's title field, and if we look up the G_BOXTEXT object type in our reference materials, we'll find that the *ob_spec* field of this type of object *does* hold a pointer to a tedinfo structure. If you look at the *ob_spec* field for this object in our listing, you'll see the value 0x0L which is obviously not a pointer. Again, this is an offset, telling us that the tedinfo for our title field is element 0 of the *rs_tedinfo[ ]* array.

The next two objects in our tree, the time and date fields, also contain offsets in the *ob_spec* member, telling us that the tedinfos for these objects are Element 2 and 3 of the *rs_tedinfo[ ]* array.

The last two objects in our tree are the OK and CANCEL buttons. They are objects of the type G_BUTTON, and if we look up these objects in our reference materials, we'll find that their *ob_spec* fields should contain, not a pointer to a tedinfo, but a pointer to the string that will be displayed in the button. Looking at the *ob__spec* fields in the button objects, we see that we once again have some offsets, 0x9L and 0xAL. These values tell us that the pointers to the button strings are found in element 9 and 10 of the *rs__strings[ ]* array.

The last four members of our object structures contain the X coordinate, Y coordinate, width and height of the objects. It's important to note that these values are given as character coordinates rather than pixel coordinates. This is because the resource construction program has no idea what resolution we're going to be running the program in. We are going to have to adjust these values so that the dialog box is drawn properly in whatever resolution we happen to be in.

Our code-resident resource tree is concluded with the array *rs__trindex[ ]*. This array will contain the addresses of each of the separate trees that make up our resource tree. In our case, we have only one tree, a dialog box, so this array contains space for only one entry. Had we had several trees—for instance, three dialog boxes—this array would have had places for the addresses of each.

If you look at the value stored in the *rs__trindex* array, you will see that, once again, we are dealing with an offset, 0L. This tells us that the address of object 0 of *rs__object[ ]* should be placed in this element of the array.

Writing a desk accessory

As I said before, writing a desk accessory is a fairly simple process. As we've done with our Time/Date utility, we can actually write the program as a normal GEM application, and then, once we've got it running, convert it to a desk accessory.

Listing 1 is just such a conversion. Most of the program is identical to last month's, and so, if you typed in last month's listing, you can use most of the code directly in this month's program. The following functions have not changed: *chk__date( )*, *set__date( )*, *get__time( )*, *get__date( )*, *get__tedinfo__string( )* and *open__vwork( )*.

Let's take a look at the function *do__acc( )*, since it is here that we set up our desk accessory, as well as initialize our dialog box. The first thing we have to do to get our desk accessory up and running is to get its name placed on the menu bar so that the user can select it. This is done with the call:

```
menu_id = menu_register (
  gl_apid, "  Name " );
```

where the integer *menu__id* is the ID number of our desk accessory returned from the call to *menu__register( )* and the integer *gl__apid* is the application ID assigned by *appl__init( )*. You don't have to worry about retrieving the value of *gl__apid* yourself; just define it as an external variable and everything else will be taken care of for you, much like the GEM global arrays.

Once we've taken care of getting our accessory registered on the menu bar, we have to initialize the dialog box. This requires replacing the offsets in the various structures with the proper pointers and changing the object coordinates and sizes from character form to pixel form.

The first thing to do is to store the address of our resource tree into the *rs__trindex[ ]* array, which is done like this:

```
rs_trindex[0] =
(long) rs_object;
```

Had we two object trees in our resource, we would have had to fill in an address for *rs__trindex[1]* too. Because we have six objects in our dialog box, the first object of a second tree (if it existed) would be the seventh object in the *rs__object[ ]* array, and we would have stored its address like this:

```
rs_trindex[1] =
(long) &rs_object[6];
```

The addresses of additional trees are stored in the same way, each address being placed in the next element of *rs__trindex[ ]* and each address being derived from the element of *rs__objects[ ]* that contains the first object in the object tree.

Now we move down one step in the hierarchy from the tree to the objects in the tree. Five of our objects require that the *ob__spec* fields be filled in with pointers. The first three—the title, time and date fields—require pointers to tedinfos. Using the offsets that the resource construction program left for us, we initialize these fields using this method:

```
rs_object[1].ob_spec =
(char *) &rs_tedinfo[0];
```

In English the above code says, "The *ob__spec* field of the second object in the array *rs__object[ ]* gets the value of a pointer to character, and that pointer points to the first tedinfo structure in the array *rs__tedinfo[ ]*." *Yikes!* I think I liked the C version better! Anyway, we initialize the other two tedinfo pointers the same way, as you can see in Listing 1.

Next we have two objects—the OK and CANCEL buttons—that must have pointers to strings placed in their *ob__spec* fields. The code that accomplishes that job (for one of the buttons) looks like this:

```
rs_object[4].ob_spec =
   rs_strings[9];
```

Once again in English, the above says, "The *ob__spec* field of the fifth object in the array *rs__object[ ]* gets the string pointer stored in Element 9 of the array *rs__strings[ ]*."

Now we have to take another step down in our hierarchy and initialize our tedinfo structures. We have three of them that need to have pointers to their *te__ptext*, *te__ptmplt* and *te__pvalid* strings filled in. We do that using a code similar to the one we used for the *ob__spec* pointers above, like this:

```
rs_tedinfo[0].te_ptext = rs_strings[0];
rs_tedinfo[1].te_ptmplt = rs_strings[1];
rs_tedinfo[2].te_pvalid = rs_strings[2];
```

The other tedinfos are taken care of in the same way.

Now all we have left to do is convert the dialog's character coordinates to pixel coordinates. Luckily, there's a function that does that dirty work for us. The call

```
rsrc_obfix ( tree_addr, object );
```

where *tree__addr* is the address of the object tree and the integer *object* is the index of the object within the tree, does all the conversions for us. In Listing 1, we've used a *for* loop to adjust all six objects with a single statement.

Waiting forever

As I said before, a desk accessory, once installed on the menu bar, waits to be called by the user. When the user clicks on the desk accessory's entry on the menu bar, the desk accessory is notified by a message from GEM. What we need to do is construct a *while* loop that'll loop "forever." In that loop we'll check to see if we've received a message to activate the desk accessory.

Once we get a message (in this case, from a call to *evnt__mesage( )*), we'll check to see if the message type (stored in *msg__buf[0]*) is an AC__OPEN message. An AC__OPEN message is sent by GEM whenever a desk accessory menu entry is clicked on by the user. When we receive this message, we then have to compare the menu ID sent to us in *msg_buf[4]* with the menu ID we obtained with our call to *menu_register()*. If they match, we go ahead and bring up our dialog box so that the user can edit the date and time.

And that's about all there is to it. I should mention that there is also a AC_CLOSE message that is sent to your desk accessory when the desk accessory is "closed." This message is important if you're using windows in the desk accessory because GEM automatically closes these windows when it returns to the desktop. Without the AC_CLOSE message, you'd have no way of knowing when to reset any window flags or other related data items you may need to update. Both the AC_OPEN and AC_CLOSE messages are defined in the gemdefs.h file.

The desk accessory link

One last note: Whenever you link a program with Megamax C, some start-up code (contained in the init.o file) is automatically linked to your object file. Desk accessories, however, have to be initialized differently when they are run, so they require different start-up codes. The desk accessory start-up code is contained in the ACC.L file supplied with the Megamax compiler. (Other compilers will have their own versions of the start-up module.) The source code for your desk accessory is compiled in the same manner as any other program, but when you get to the link step, you must be sure to select ACC.L as the first file in your link list. If you don't, your desk accessory will not run.

C-
MANSHIP

C-manship Listing 1-C

```
/*********************************************************/
/*               C-manship, Listing 1                    */
/*                    ST-Log #24                         */
/*              Developed with Megamax C                 */
/*********************************************************/
#include <osbind.h>
#include <gemdefs.h>
#include <obdefs.h>
#include "date.h"

#define TRUE   1
#define FALSE  0
#define MATCH  0

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

extern int gl_apid; /* Global application ID. */

int handle,   /* Application handle. */
    dum,      /* Dummy storage.      */
    menu_id;  /* Our accessory's ID. */

char *get_tedinfo ();

int msg..buf[8]; /* Message buffer. */

OBJECT *datedial_addr; /* Pointer to dialog box. */

/*****************************************************
*   Resource tree
*****************************************************/

char *rs_strings[] = {
"DATE AND TIME",
"",
"",
"000000AM",
"Time: __:__:__ __",
"999999AA",
"000000",
"Date: __/__/__",
"999999",
"OK",
"CANCEL"};

TEDINFO rs_tedinfo[] = {
0L, 1L, 2L, 3, 6, 2, 0x1180, 0x0, -1, 14,1,
3L, 4L, 5L, 3, 6, 0, 0x1180, 0x0, -1, 9,18,
6L, 7L, 8L, 3, 6, 0, Øx1180, 0x0, -1, 7,15};

OBJECT rs_object[] = {
-1, 1, 5, G_BOX, NONE, 0x30, 0x21121L, 0,0, 41,12,
2, -1, -1, G_BOXTEXT, NONE, SHADOWED, 0x0L, 2,1, 37,1,
3, -1, -1, G_FTEXT, EDITABLE, NORMAL, 0x1L, 12,3, 17,1,
4, -1, -1, G_FTEXT, EDITABLE, NORMAL, 0x2L, 13,5, 14,1,
5, -1, -1, G_BUTTON, 0x5, SHADOWED, 0x9L, 2,9, 17,1,
0, -1, -1, G_BUTTON, 0x27, SHADOWED, 0xAL, 22,9, 17,1};

long rs_trindex[] = {
0L};

#define NUM_OBS 6

/*****************************************************
* Main program.
*****************************************************/
main ()
{
   appl_init ();          /* Init application.          */
   open_vwork ();         /* Open virtual workstation. */
   do_acc ();             /* Go do our thing.           */
}

/*****************************************************
* do_acc ()
* Initialize and handle desk accessory.
*****************************************************/
```

```
do_acc ()
{
   int x;   /* Loop variable. */

   /* Place our accessory on the menu bar. */
   menu_id = menu_register ( gl_apid, "  Date/Time " );

   /* Initialize resource tree. */
   rs_trindex[0] = (long) rs_object;
   datedial_addr = (OBJECT *) rs_trindex[0];
   rs_object[1].ob_spec = (char *) &rs_tedinfo[0];
   rs_object[2].ob_spec = (char *) &rs_tedinfo[1];
   rs_object[3].ob_spec = (char *) &rs_tedinfo[2];
   rs_tedinfo[0].te_ptext = rs_strings[0];
   rs_tedinfo[0].te_ptmplt = rs_strings[1];
   rs_tedinfo[0].te_pvalid = rs_strings[2];
   rs_tedinfo[1].te_ptext = rs_strings[3];
   rs_tedinfo[1].te_ptmplt = rs_strings[4];
   rs_tedinfo[1].te_pvalid = rs_strings[5];
   rs_tedinfo[2].te_ptext = rs_strings[6];
   rs_tedinfo[2].te_ptmplt = rs_strings[7];
   rs_tedinfo[2].te_pvalid = rs_strings[8];
   rs_object[4].ob_spec = rs_strings[9];
   rs_object[5].ob_spec = rs_strings[10];

   /* Set all the objects' coordinates. */
   for ( x=0; x<NUM_OBS; ++x )
      rsrc_obfix ( datedial_addr, x );

   /* Wait forever for messages. */
   while ( 1 ) {
      evnt_mesag ( msg_buf );

      switch ( msg_buf[0] ) {   /* msg_buf[0] is message type. */

         /* Open our accessory. */
         case AC_OPEN:
            if ( msg_buf[4] == menu_id )
               do_date ();
      }
   }
}


/*****************************************************
* do_date ()
* Loads the resource file and handles the dialog box.
*****************************************************/
do_date ()
{
   int dial_x, /* Dialog's X coord.  */
       dial_y, /* Dialog's Y coord.  */
       dial_w, /* Dialog's width.    */
       dial_h, /* Dialog's height.   */
       choice, /* Exit button clicked from dialog.      */
       okay;   /* Flag indicating if entered date valid. */


   char date_str[8],  /* String to hold date. */
        time_str[10]; /* String to hold time. */

   char *string; /* Temp string pointer. */


   graf_mouse ( ARROW, 0L );

   get_time ( time_str );
   get_date ( date_str );

   /* Copy system time and date into dialog box, */
   string = get_tedinfo_str ( datedial_addr, TIMEFLD );
   strcpy ( string, time_str );
   string = get_tedinfo_str ( datedial_addr, DATEFLD );
   strcpy ( string, date_str );

   /* Prepare dialog box for drawing, and init flag. */
   form_center ( datedial_addr, &dial_x, &dial_y, &dial_w, &dial_h );
   form_dial ( FMD_START, 0, 0, 10, 10, dial_x, dial_y, dial_w, dial_h );
   okay = TRUE;
```

```
    /* This loop repeats until the user clicks the CANCEL button       */
    /* or until the user enters a valid date and clicks the OK button. */
    do {
       /* Draw dialog and allow user to manipulate it. */
       objc_draw ( datedial_addr, 0, 8, dial_x, dial_y, dial_w, dial_h );
    choice = form_do ( datedial_addr, TIMEFLD );

    /* Reset the state of the chosen button. */
    datedial_addr[choice].ob_state = SHADOWED;

    /* If OK button clicked, check entered date and set system */
    /* date if date entered is valid,                          */
    if ( choice == OKBUTN ) {
       okay = chk_date ( datedial_addr );
       if ( okay )
          set_date ( datedial_addr );
    }
 }
 while ( okay == FALSE && choice == OKBUTN );

 /* Get rid of the dialog box. */
 form_dial ( FMD_FINISH, 0, 0, 10, 10, dial_x, dial_y, dial_w, dial_h );



*****************************************************
 chk_date ()
 Examines the strings in dialog for a valid date
 and valid time.
****************************************************/
hk_date ( dial_addr )
BJECT *dial_addr; /* Address of dialog box. */

  int mnth, day, year, /* Date and time broken down into integer portions.*/
      hour, min, sec,
      space,           /* Flag indicating non-valid chars in time string. */
      okay,            /* Flag indicating valid time and date. */
      x;               /* Loop variable. */

  char m[3], d[3], y[3], /* Date and time as character arrays. */
       h[3], mn[3], s[3],
       ap[3];              /* "AM" or "PM" */

  char *date_str, /* Pointer to string containing date. */
       *time_str; /* Pointer to string containing time. */

 /* Init date and time integers to error condition. */
 mnth = day = year = hour = min = sec = -1;

 /* Get address of string containing date. */
 date_str = get_tedinfo_str ( dial_addr, DATEFLD );

 /* Convert date string to integer format. */
 if ( strlen ( date_str ) == 6 ) {
    strncpy ( m, date_str, 2 );
    m[2] = 0;
    strncpy ( d, &date_str[2], 2 );
    d[2] = 0;
    strncpy ( y, &date_str[4], 2 );
    y[2] = 0;
    mnth = atoi ( m );
    day = atoi ( d );
    year = atoi ( y );
 }

 /* Get address of string containing time. */
 time_str = get_tedinfo_str ( dial_addr, TIMEFLD );

 /* Check for spaces in time string. */
 space = FALSE;
 for ( x=0; x<6; ++x )
    if ( time_str[x] == ' ' )
       space = TRUE;

 /* Convert time string to integer format. */
 if ( (strlen ( time_str ) == 8) && !space ) {
       strncpy ( h, time_str, 2 );
       h[2] = 0;
       strncpy ( mn, &time_str[2], 2 );
       mn[2] = 0;
       strncpy ( s, &time_str[4], 2 );
       s[2] = 0;
       hour = atoi ( h );
       min = atoi ( mn );
       sec = atoi ( s );
       strcpy ( ap, &time_str[6] );
 }

 /* Examine time and date for validity. */
 if ( mnth < 1 | mnth >12 | day < 1 | day > 31
```

C-MANSHIP

CircuitMaker

CircuitMaker is a professional full featured program that enables you to design, construct and test an unlimited variety of digital circuits. Using CircuitMaker, you eliminate the need to purchase breadboards, integrated circuits, wire and power supplies. CircuitMaker allows you to design and test your digital circuits with just a few clicks of the mouse!

CircuitMaker is designed for the professional as well as the student that is just learning about digital logic. CircuitMaker is a must for your electronic projects!

Only $79.95

iliad Software Inc.

P.O. Box 1144
495 West 920 North
Orem, Utah 84057
(801) 226-3270
Office hours 10:00AM-6:00PM MST

CIRCLE #123 ON READER SERVICE CARD.

Magnetic Images™

CLIP ART !

Quality Images for use with:
- Supercharged EasyDraw
- Fleetstreet Publishers
- Publishing Partner
- First Word Plus
- Calamus

Disk 1 : Computer, Office, Music, School, Travel, Trans.
Disk 2 : Business, Sports, Animals, Party, Religious
Disk 3 : Food, Borders, Medicine, Old West, Newsletter
Disk 4 : Hands, Seasons, Pirates, Tools, Personal, America
Disk 5 : Theater, Corners, Zoo, Menu, Outdoor
Disk 6 : Ad man's Special: Computer Products

$19.95 **per disk** (Add $2.50 P & H per order)

Magnetic Images Co.
P.O. Box 17422, Phoenix, AZ 85011
(602) 265-7849
Dealer inquiries welcome.

CIRCLE #124 ON READER SERVICE CARD.

Hand in hand *(from page 97)*

the novice user.

Why is Atari's name absent from the list of computer products that these stores are carrying? Good question. It could be Atari's lack of advertising. For the last three years or so (since Jack Tramiel bought the company, come to think of it), Atari has done little or no advertising for any of their computer products.

Not to create a chicken-egg problem here, but the reason stores are not carrying Atari products may be because users don't want them. As of March 1987, about 16% of U.S. households had personal computers. Today, according to *Computer & Software News'* Annual Home Market Study, about 19% of households have computers. According to this same study, Commodore is the brand of PC most likely to be found in the home, followed by Apple, IBM, Tandy, Texas Instruments and Atari.

Comparing 1987 and 1988 figures, Atari dropped from a household share of 6% to 3%. The only machine found less often in homes is the Coleco Adam, at 1% in both 1987 and 1988. According to statistical analysis, conclusions cannot be reliably drawn from data in the 1% to 5% range. The exact amount depends upon the number of people responding to the survey, but this means that the error in data gathering could be large enough to obscure the data itself.

Of even more interest to us and the department stores who are selling PCs is the brand of computer that people are most likely to buy. In March 1987, 1% of the survey respondents said they were planning to buy an Atari computer. In May 1988, 1% also said they were planning to buy an Atari computer. Apple and IBM were the brands reported to be the most likely to be purchased. In fact, over the course of the year, Apple gained share while IBM lost it.

What is going on here? People say they don't plan to buy an Atari computer so the stores don't sell them. The stores don't sell Atari computers because folks say they don't plan to buy Atari. In either case, the result is the same. Atari's share of the U.S. market is dwindling while other computer manufacturers succeed. At least the Atari ST computers are hot in Europe.

Being an Atari computer user is never an easy task, but it sure is fun. See you next month.

```
         | year < 0 | year > 99 | hour < 0 | hour > 23 | min < 0
          | min > 59 | sec < 0 | sec > 59 |
         ( (strcmp (ap,"AM")!=MATCH) && (strcmp (ap,"PM")!=MATCH) ) ) {
      okay = FALSE;
      form_alert ( 1, "[1][Date or time not valid!][CONTINUE]" );
   }
   else
      okay = TRUE;

   return ( okay );
}

/****************************************************
* set_date ()
* Sets the system time and date to the values
* entered into the dialog box.
****************************************************/
set_date ( dial_addr )
OBJECT *dial_addr; /* Address of dialog box. */
{
   char *string; /* Temporary string pointer. */
   char s[3];    /* Temporary string storage. */
   int h,        /* Work variable.            */
       time,     /* Time in system format.    */
       date;     /* Date in system format.    */

   /* Get address of string containing time. */
   string = get_tedinfo_str ( dial_addr, TIMEFLD );

   /* Extract "hours" portion and convert to integer. */
   strncpy ( s, string, 2 );
   h = atoi ( s );

   /* Adjust hour to the 24-hour clock format. */
   if ( (strcmp ( &string[6], "PM" ) == MATCH) && (h != 12) )
      h += 12;
   if ( (strcmp ( &string[6], "AM" ) == MATCH) && (h == 12) )
      h = 0;

   /* Shift bits into the proper position and place them */
   /* into the time integer.                             */
   h = h << 11;
   time = h;

   /* Get the "minutes" portion, convert to integer,     */
   /* shift bits and place them into the time integer.   */
   strncpy ( s, &string[2], 2 );
   h = atoi ( s );
   h = h << 5;
   time = time | h;

   /* Process the "seconds" portion of the time. */
   strncpy ( s, &string[4], 2 );
   h = atoi ( s ) / 2;
   time = time | h;

   /* Set the system clock to the new time. */
   Tsettime ( time );

   /* Get the address of the string containing the date. */
   string = get_tedinfo_str ( dial_addr, DATEFLD );

   /* Process the "month" portion. */
   strncpy ( s, string, 2 );
   h = atoi ( s );
   h = h << 5;
   date = h;

   /* Process the "day" portion. */
   strncpy ( s, &string[2], 2 );
   h = atoi ( s );
   date = date | h;

   /* Process the "year" portion. */
   strncpy ( s, &string[4] );
   h = atoi ( s ) - 80;
   h = h << 9;
   date = date | h;

   /* Set the system to clock to the new date. */
   Tsetdate ( date );
}

/****************************************************
* get_time ()
* Gets system time and converts it to string format.
****************************************************/
get_time ( string )
char *string; /* Pointer to string in which to store time. */
{
   int time,              /* Time in system format.              */
       hour, min, sec;    /* Time broken down into separate ints. */

   char s[3];             /* "AM" or "PM" */

   /* Get system time and break down into individual components. */
   time = Tgettime ();
   sec = ( time & 0x001f ) * 2;
   min = ( time >> 5 ) & 0x003f;
   hour = ( time >> 11 ) & 0x001f;

   /* Convert system 24-hour format to regular 12-hour format. */
   if ( hour > 11 ) {
      strcpy ( s, "PM" );
      if ( hour > 12 )
         hour -= 12;
   }
   else {
      strcpy ( s, "AM" );
      if ( hour == 0 )
         hour = 12;
      }

      /* Convert and add hours to time string. */
      if ( hour < 10 ) {
         string[0] = '0';
         sprintf ( &string[1], "%d", hour );
      }
      else
         sprintf ( string, "%d", hour );

      /* Convert and add minutes to time string. */
      if ( min < 10 ) {
         string[2] = '0';
         sprintf ( &string[3], "%d", min );
      }
      else
         sprintf ( &string[2], "%d", min );

      /* Convert and add seconds to time string. */
      if ( sec < 10 ) {
         string[4] = '0';
         sprintf ( &string[5], "%d", sec );
      }
      else
         sprintf ( &string[4], "%d", sec );

      /* Add "AM" or "PM" to time string. */
      strcpy ( &string[6], s );
   }

/*****************************************************
 * get_date ()
 * Gets system date and converts it to string format.
 *****************************************************/
get_date ( string )
char *string; /* Pointer to string that will contain the date. */
{
   int date,             /* Date in system format.      */
       day, mnth, year; /* Date broken into components. */

   /* Get system date and convert to individual components. */
   date = Tgetdate ();
   day = date & 0x001f;
   mnth = (date >> 5) & 0x000f;
   year = ((date >> 9) & 0x007f) + 80;
   year = year % 100;

   /* Convert and add "months" portion to date string. */
   if ( mnth < 10 ) {
      string[0] = '0';
      sprintf ( &string[1], "%d", mnth );
   }
   else
      sprintf ( string, "%d", mnth );

   /* convert and add "days" portion to date string. */
   if ( day < 10 ) {
      string[2] = '0';
      sprintf ( &string[3], "%d", day );
   }
   else
      sprintf ( &string[2], "%d", day );

   /* Convert and add "year" portion to date string. */
   sprintf ( &string[4], "%d", year );
}

/*****************************************************
 * get_tedinfo_string ()
 * Returns a pointer to an editable string in a
 * dialog box.
 *****************************************************/
char *get_tedinfo_str ( tree, object )
OBJECT *tree; /* Address of dialog box.        */
int object;   /* Object that contains the string. */
{
   TEDINFO *ob_tedinfo; /* Pointer to a tedinfo structure. */

   ob_tedinfo = (TEDINFO *) tree[object].ob_spec;
   return ( ob_tedinfo->te_ptext );
}

/*****************************************************
 * open_vwork ()
 * Opens a virtual workstation.
 *****************************************************/
open_vwork ()
{
   int i;

   /* Get graphics handle, initialize the GEM arrays and open  */
   /* a virtual workstation.                                   */

   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 );
}
```

C-manship
Listing 2-Date. H

```
#define DATEDIAL 0
#define TIMEFLD 2
#define DATEFLD 3
#define OKBUTN 4
#define CANBUTN 5
```

MOVING?

DON'T MISS A SINGLE ISSUE

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

DO YOU HAVE A QUESTION ABOUT YOUR SUBSCRIPTION?

Check the appropriate boxes below:
- ☐ New subscription. Please allow 4 to 8 weeks for your first copy to be mailed.
- ☐ Renewal subscription. Please include a current address label to insure prompt and proper extension.
- ☐ 1 year—$28.00. This rate limited to the U.S. and its possessions.
- ☐ Payment enclosed. ☐ Bill me.

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

Name ____________
Street Address ____________
City ____________ State ______ Zip ______

ATTACH LABEL HERE

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

C-MANSHIP
E N D

ST GOSSIP from Hollywood, USA

by TG

What game machine image?

As Sherlock Holmes would say, "Quick, Watson, the game's afoot." Well, in this case, the game is an ST . . . a **68000-based game machine** to be exact. Atari is pushing a "game version" of the ST system for Christmas. How does a game version of an ST differ from the version you and I have been using for the last three years? For one, it doesn't have a disk drive, a monitor or, we hope, the ST name on the case.

Lots of people in the industry are curious about this move, since one major complaint among the dealers who are trying to sell the Atari to the business world is the company's "Game Machine Image." One group of people who are sure to appreciate the move, though, are the stockholders. Game machines are still *hot*, and an ST can blow the doors off some of the smaller units like the Nintendo and Sega. Of course, we all know that Atari feels it's in the toy business for the quick money, and it's in the computer business for the long-term profit. This leads us to think about. . .

Game machines are still *hot*, and an ST can blow the doors off some of the smaller units like the Nintendo and Sega.

How did they do that?

Six months ago Atari was at its lowest point in years in terms of user confidence. Now, after six months of management changes, fighting off the chip shortages, postponing the new operating system, weeding out the weak dealers and putting the final touches on the mail-order shutdown, Atari has started to **reestablish confidence** in the minds of the consumer. This certainly had to be one of the most difficult periods in Atari's history, and yet they emerged from it in better shape than when they started. The new power of GDOS has certainly had a lot to do with it, and I think people are starting to feel that Atari does indeed have a long-range plan, one that we are not privileged to share, but a long-term plan nonetheless.

Unless I miss my guess there will be some **exciting specials** from Atari for the Christmas season. Atari has been bringing some tried and true retailing plans to the computer area for the last year or so and they are starting to have some affect on its market share.

Random notes

How about that **upgrade plan** for single-sided drive owners? What do you mean, "What upgrade plan?" Oops, guess that one's out of the bag. See your dealer for details in the next 30-60 days.

Look for companies like Supra and ICD to be **dropping the price** of their hard disks soon. An informal survey of 25 Atari dealers shows that better than 35% are building and selling private-label drive systems, with 40-meg systems selling for less than the price of the "brand name" 20-megs.

I'm sure that most of you have been checking **Atari's stock price** now and then. Has anyone figured out how much you would have made if you had bought it when it first came out and again just after the market crash of early this year? Or how much you could have lost buying it at its high before the crash? This stock has provided more action than the average army barracks poker game. If nothing else, this fact alone should make it a stockbroker's nightmare.

Now that **GenLock** is really available, look for a high-quality picture grabber to be released within the next 30 days. The product is done and only awaits the availability of low-cost RAM chips.

With fewer, but stronger, Atari dealers in place across both coasts, it looks like Atari is ready to start the Big Push into the **American business market**. The software base is clearly better than it was just a few months ago, and the dealers who are left are the strongest of the crew that started this long, long summer.

Now it's time to see if Atari has anything up its sleeve for 1989. You can look to the next three months to serve as a guide to the **future of Atari** throughout 1989. If the plans I've heard about actually come to pass, you might not recognize Atari one year from now. The guys at Sunnyvale have been acting more and more like their European counterparts for the last six months, quietly calling in dealers from around the country to meetings at the corporate headquarters to lay plans for what they hope will finally be the Big Push. Time will tell, but the simple fact they have been consulting with dealers marks a major change in the corporate thinking at Sunnyvale. Now, if they would just continue to beef up the developer support program and start a little institutional advertising. . . .

. . . three year cycles

Many of those who have watched Atari for the last three years, and Commodore before that, realize that JT-controlled companies seem to follow three-year product cycles. We expect that there will be **major changes to the Atari product lineup** within the next months. It's highly unlikely that any products will be dropped, but I think you can look forward to major new products being released.

The advantage Atari offered to the typical user in 1985 was a chance to buy true state-of-the-art equipment at a price real people can afford. I believe the time has come (in the mind of Atari) to push the state of the Atari systems forward again. Each time Atari—or in the past, Commodore—did this it picked up a year of free publicity.

Now it's time to see if Atari has anything up its sleeve for 1989. You can look to the next three months to serve as a guide to the future of Atari throughout 1989.

You may remember when the newly purchased Atari Corp. unveiled the 520ST to an unsuspecting world back in 1985. The company received free ink in every computer magazine and most newspapers in the free world for the next three months. Industry people clucked and commented on this new, powerful, low-cost system that seemed to spring up from nowhere. Putting the money that other companies spend on advertising into product development can result in **self-promoting products**—*if* you can bring in results on a regular basis and *if* you can keep your product development under wraps until you release the completed project upon an unsuspecting world. If Atari is going to do it again, then it will be in the next six months.

DATABASE DELPHI

by Andy Eddy

I've been preparing for hibernation, with the weather getting ready to slide to the colder side; so I've been somewhat preoccupied. Would you mind turning the cover towards me so I know what this issue is about? Ah, yes, games. Isn't that *special.* Well, what shall we talk about then? How about, oh. . .*games*?!

No "E" ticket required

There are two methods for playing games with your ST and DELPHI. (Actually, the games on DELPHI aren't ST-specific, so any computer user who can get online can play them.) The first comes in the form of the resident gameware that DELPHI has in place. They include games like sports simulations, text adventure contests and a Star Trek game. (What game library would be complete without a Star Trek game?)

Simply type *GO ENT* and you'll be brought to the Entertainment and Games menu. To show the varied offering, here's a glance at the menu selections:

```
Entertainment & Games Menu:

Adventure Games         Music Reviews
Astro-Predictions       Poll
Board Games             Role Playing Games
Collaborative Novel     Sports Games
DELPHI Casino           Trivia Quest
GameSIG                 VT Games
Logic Games             Witt's End (Misc games)
Members' Choice         Help
Movie Reviews           Exit

FUN>Which Selection?
```

At this bill of fare, you can type the first few letters of your choice. (Type enough to make your selection unique so DELPHI will know exactly what you want to do.) Once there, you have even more to pick from.

You might have noticed from the main entertainment menu that DELPHI has a GAMESIG (run by SCORPIA and ALCHEMIST1; to maintain the mystique, they've chosen to keep their names secret). Just like any other SIG (Special Interest Group), you can read messages in the Forum—or post your own to others—scan a database of reviews and hints, and more. Again, it's located off the Entertainment menu by typing *GAM* from the FUN prompt.

Watchin' ware

The other way to enjoy gaming on the ST via DELPHI is the database in the ST SIG. There was a time when public domain software was generally inferior to commercial products, but with dazzling color palettes to work with and the increase in home-brewed programming efforts, some of the freeware and shareware programs out now can be considered market quality.

To get to the games database area, type *DAT GAM* (an abbreviation of DATABASE GAMES) at the Atari ST prompt. This will bring you right up to where the action is. At the time of this writing, there were over 170 files of game programs and demos in the Games and Entertainment section.

There was a time when public domain software was generally inferior to commercial products, but with dazzling color palettes to work with and the increase in home-brewed programming efforts, some of the freeware and shareware programs out now can be considered market quality.

You could type *DIR* to begin a directory listing (or *DIR NS*, which will give you the entire directory without any "More?" pauses), or *REA* to view the descriptions in chronological order, starting with the most recent file. The file descriptions display information such as the size of the file and how many times it's been downloaded, as well as a brief piece of text that explains something about the program. The file description looks like this:

```
Name: BASEBALL.ARC
Type: PROGRAM
Date: 22-FEB-1988 19:02 by OLY
Size: 112640 Count: 18

This Arc'd file contains a baseball game in compiled LDW
BASIC.  The Game is called Statistically Accurate Baseball, a
shareware game by Joe Damore.  While this game is not heavily
dependent on arcade style graphics, it is more dependent on
strategy.  All players and teams are 100 percent accurate
(hence the name). This game will intrigue you and keep you at
it for hours on end.  To play game after unarcing, double
click on the file SAB87.PRG.  But be sure to read the doc
file first.

Keywords: STRATEGY, BASEBALL, ACCURATE, ARC, COMPILED
```

The games in the DBs (as we call the databases) are well worth the downloading time. When you consider the capability of the ST, given the larger RAM space to work with, some of the programs can get quite large after the addition of digitized sound and flashy graphics.

Hunting for game

Baseball fans have a simulation game for their favorite sport. Called *Statistically Accurate Baseball* (SAB), written by Joe Damore, it comes with statistical data for four classic, historical teams, who, for the most part, play mathematically correct, based on the actual player averages and pitching stats. Playing is accomplished by clicking on the desired offensive and defensive strategy—bunt, infield in, stealing a base and more. While not equipped with arcade-like graphics, the results of your managerial decisions are described to you as a running play-by-play. Stats can be printed out after the game is over, in order to track the teams' progress.

If you pay the $12 shareware donation, which Damore claims will cover his "three years of hard work, shipping and a disk," you'll receive 32 more team files. *SAB*, found in the Games section of the ST SIG as BASEBALL.ARC, runs in color or mono.

Eric Lindros is a busy guy, with two maze adventures on our list. *Escape* (ESCAPE.ARC) and *MegaMaze* (MEGAMAZE GAME) are intense strategic battles where the main goal is to collect treasures and fend off adversity—a fairly normal gaming premise. They will greatly challenge your adventuring skills. These games are playable in color or mono. A $5 shareware fee is requested for each game. Beyond that, Lindros also is holding a contest for improvements to his programs. The winner, whose ideas Lindros feels are the best, will get $50 and credit in the upgraded version.

One thing to be careful of with *Escape* and *Megamaze* is that between the two ARCs (a file type that I'll discuss for the uninitiated at the end of this article) that hold the necessary files, there are some identically named files. Be sure to either extract each ARC to a different disk or folder, so there's no potential of a file name conflict.

Lastly, most everyone remembers a commercial climb-and-chase game from a while ago called *Lode Runner*, by Broderbund. Well, WMS and KKS (as they are listed on the game's title screen) have created a monochrome ST clone called *Runner's Revenge*, that brings back all of the intrigue and lively game play of the original.

One thing to keep in mind is that *Runner's Revenge* is over 40K in size ARCed, which extracts out to 70 files and more than 87K, so be ready with lots of open disk space. Additionally, the software requires you to place certain files in specifically named folders (such as .SND files going into a folder called SOUNDS). The best plan is to first extract the README.TXT file, then follow the instructions by creating the necessary folders beforehand. When you have that all done, you can extract the proper files to their respective locations.

Not Noah's ARC, silly

We've made mention of files with the ARC extender before, but I haven't explained it to those users who are unaware exactly what makes them so special.

The extender ARC is derived from *archive*, which in this case is a file—or in some cases, a group of files—compressed into one smaller file. The benefit of an ARC file to the telecomputing enthusiast is obvious: the shorter the file, the less time online downloading it. This may also enable storage of the multiple ARC(s) on fewer floppies or on less hard-drive space.

But being able to ARC files has another plus: Related files can be grouped together, so there is no confusion to what data file belongs to what program. Similar to the folder concept of GEM on the ST, you can keep together all files that work together. For example, many programs come with a README.DOC file, to explain how to set up the program or to describe changes that have been made from earlier versions; other software that requires graphics (like backgrounds for a game) may have multiple DEGAS screens associated with it. With the ARC process, they can be stored *en masse*, under one roof, so to speak.

Converted to the ST by Harvey Johnson, the ARC.TTP file is the one that handles all these duties, and is even compatible with most versions of ARC for other computers! To make it even simpler to use, Charles Johnson's ARCSHELL (upgraded to an even higher level since its ST-Log debut) is without comparison. Most of the files in the DBs are ARCed,, so if you intend to do any downloading, these two files should be the first ones to acquire.

To get them, go to the Applications section (type *DAT APP*) and type *READ ARC UTILITY* to get ARC.TTP. Then go to Utilities (type *SET UT*, from the DBASES:App > prompt) and type *READ ARC SHELL 1.95* (the latest version at this writing). Without question, these two files will be valuable assistants to your file transfer and/or storage situation.

Well, it's about time for me to crawl into my cave for another month. Till next month, C U online. . . .

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.

TUTORIAL

STEP 1 Playing Games with Graphics

by
Maurice
Molyneaux

Not just a game machine!

True, true, STs are not just game machines, but there are a lot of games available for them. I'm not much of a game player. I rarely get involved in playing game software. . . but I've had my hand in designing the graphics for a few. To me that's more interesting than playing the game proper. Most of you probably wouldn't agree on that point, but that's just individual taste. Anyway, in keeping with the new approach to "Step 1" I described last month, this issue I'll begin discussing using your ST for a task many of you are no doubt interested in: designing game graphics.

Before you wonder; no, I've never actually programmed a game in my life. To paraphrase Doctor Leonard McCoy, "I'm an artist, not a programmer!" What games have I designed graphics for? Let's see, there was a Star Trek game that unfortunately never got off the ground, Omnitrend's games *Paladin* and *Universe III*, in addition to *Breach*. "*Breach*?" I hear some of you asking. I'm not responsible for the graphics on the ST version of the game, but I did the Amiga version. Now, don't go scream "traitor." I never once touched an Amoeba (er, you know what I mean) in creating these graphics (I was contacted too late for the ST version. . .too bad). In fact, on both *Paladin* and *Universe III*, I also created the graphics for the IBM versions, as well, without touching a PC. The graphics for all versions were designed on my trusty ST.

Pigeonholing

I'm not fond of generalizing and classifying things by types (because that tends to lead to stereotyping), but sometimes a little pigeonholing can be useful to define what you're working with. There are essentially two types of display methods used with computers—vector and raster—and these break down the two most common categories of game graphics as well.

Vector graphics are those which consist of lines plotted between points. Look at an arcade *Asteroids* or *Star Wars* game. Notice how everything is drawn by lines? Notice that there are no solid (filled in) areas, and the lack of obvious pixels and jaggies on these vector displays? This is all because vector-plotted graphics consist entirely of lines plotted between points. All the objects are drawn by defining a series of points and drawing connecting lines between them. This gives great flexibility in that the objects can be resized or turned every which way without losing any detail. The disadvantage is that, generally, by using this method you can't get much detail anyway. (Imagine the game trying to draw a metallic starship—in real-time—totally by vectors! Without powerful, expensive hardware, you can't do it well.)

Raster graphics are those graphics defined by dots on a finite grid. Look at your ST's screen. Each letter is composed of dots, arranged in such a fashion that they look like something legible. Examples of arcade games using raster images range from *Pac-Man* to *AfterBurner*. Raster images are usually more colorful than vector graphics, with lots of small details and areas of solid color, not unfilled outlines. The problem here is that of dot size. The bigger they are, the more "jaggies" you see (particularly when there are lines that are

There are essentially two types of display methods used with computers—vector and raster—and these break down the two most common categories of game graphics as well.

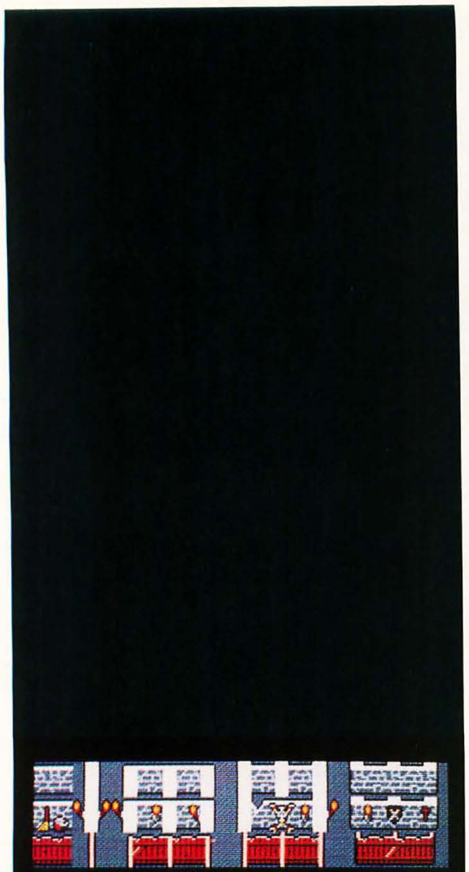


Figure 1

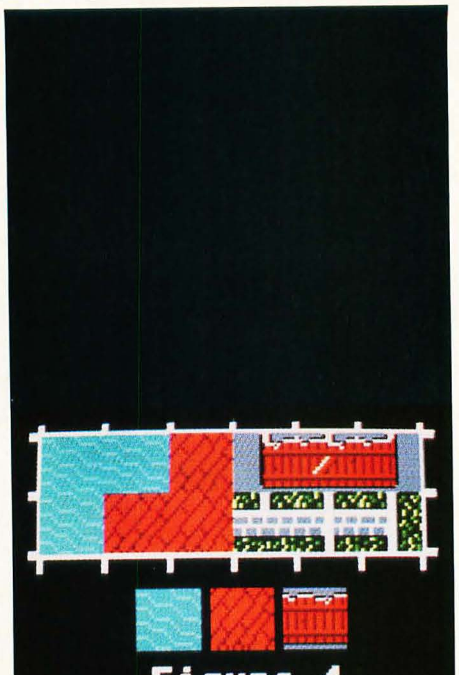


Figure 4

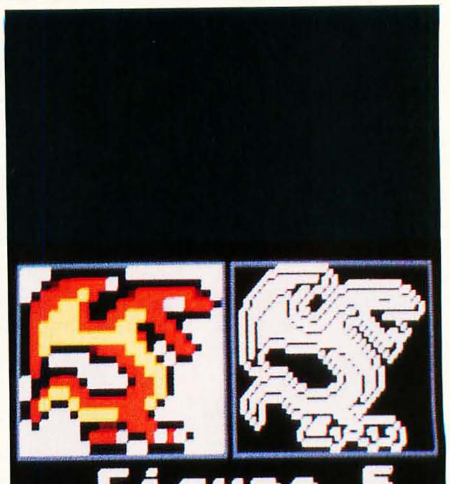


Figure 5

A mask is a single-color element, shaped like the object it is assigned to, but often a pixel or so larger.

not perfectly vertical or horizontal).

Taking all this down to the level of your ST, there are both vector-plotted and raster-type graphics used, but because of the ST's construction, everything becomes a raster image before it is displayed on the screen. *Starglider* uses vector-plotted graphics but, since the ST's video chip doesn't create the screen display using rasters, we get "jaggied" vectors. What happens in the ST is that the vector coordinates are plotted on the computer's "grid map" of the screen, and lines are drawn between these points. Since the ST's graphics output is raster (320x200, 640x200, or 640x400), these lines must be drawn by plotting pixels on the screen, resulting in jaggies. This is the reason the graphics on games like *Starglider* don't look as slick as those arcade vector graphics (which use special vector plotting hardware). The advantage on the ST is that once the vector lines have been plotted, a fill of color can be aimed inside the shape, making it solid. To create solid-looking objects with vectors requires lines plotted so close together that they create the illusion of solid color.

Laying tile

All of the games of Omnitrend's that I have designed graphics for have used what are known as "tile" graphics. Tile graphics refers to building up images out of numerous small square, graphic fragments, which are laid down in patterns just like tile. Each individual "tile" has a distinct picture on it, like grass, dirt, bushes, a segment of wall, a vending machine, etc. Thus, by arranging a number of wall tiles, you could make a building. Cluster a bunch of grass tiles and surround them with bush tiles, and you have a meadow or lawn, etc.

In many cases, there are numerous related tiles, such as walls at different angles (horizontal, vertical, corner sections, wall intersections, etc.) For *Paladin*, I designed a whole range of wooden and stone, wall segments, including a few duplicates with details added (Figure 1).

In many games, like the Ultima series, everything is represented by tiles, from castles to characters. When your character moves over a square of grass, you no longer see that square, but the tile representing your character or party. In games like those of Omnitrend's, there are two distinct types of tiles, known as "terrain squares" and "object squares." Terrain squares are just like those defined above, tiles of background material: walls, trees, etc. Object squares contain things that go over terrain squares, like characters and/or objects you can pick up. So, when you move your soldier onto a square occupied by a table, you can see the table behind him.

Of course, this is not to say that every single item in the game that is not a wall or floor, or every single person shown, is an object square and not a terrain square. In Paladin there are a number of human figures ("extras," I like to call them) which are, in fact, terrain squares. Why? Well, while it is necessary for some objects to be relocated, taken and even dropped elsewhere, some things in the game really shouldn't be moving around.

The advantage to the first technique, where all tiles are in essence terrain squares, is that it is simple to do. The programmer merely "stamps" copies of certain tiles on the screen in a grid pattern. When your character moves onto a given square, the tile there is replaced by the tile of your character. No fuss, no muss.

The trouble with that system is that the player can't see what his character is on top of (if he's forgotten). This is where the terrain and object tile system has its advantages, because it allows you to see past objects to what they're on top or in front of (so to speak.) Furthermore, if handled correctly, it can make the tile boundaries seem a little less rigid. In Omnitrend's games, when you move one of your characters from one square to another, you actually see the character step from one to the next while the screen scrolls.

The disadvantage is that it is a bit more complicated to do this. The programmer can't merely replace one tile with another, but must place the object square over the terrain square. In Omnitrend's games, this necessitates the graphic designer to create masks for each object. A mask is a single-color element, shaped like the object it is assigned to, but often a pixel or so larger. When the program must place an object over a terrain square, it can't merely just stamp it down, because it would just wipe out the background (like in a terrain-square-only approach.)

The method needed is some way to intermix the terrain and object squares without one totally wiping out the other. Objects tend not to fill up an entire square, leaving a lot of blank space. This blank space is usually left as the computer's background color (color 0), thus the programmer could execute a Logical OR operation, whereby the pixels of the background color that appear in the object square would not be plotted over the terrain square, letting it show through.

The trouble with this is that the colors may intermingle. Imagine you have two transparencies; on one is some grass, on the other is a man. Slap these two into a projector at the same time. Where the man overlaps the grass you get a yucky-looking mixture. This is what would happen if the programmer used a Logical OR alone. Furthermore, since in this method the background color is treated as utterly transparent, you can't use the background color in your object because that part will become see-through to the background. This is a real problem, considering the limited numbers of colors available on most personal computers.

The solution is to prepare the terrain square to receive the object. This is where the mask comes in. The mask is stamped down first, in essence cookie-cutting a hole into the terrain square. The object is then Logical ORed onto the square, right over the hole made by the mask.

The reason for having a mask that is larger than the object is to make certain a given object will show up over all types of terrain. Say you had a soldier in a green uniform as your object. The terrain square the object is on contains green grass. Green on green would be hard to see. The overlapping border of background color (usually white or black—though I prefer black because white tends to wash out other colors) keeps the object distinct from the terrain.

Let's get graphic

Okay, we've covered how tile graphics work and two variations on them. What about actually drawing them?

The scale of the tiles must be established before drawing begins. In the case of Omnitrend's ST games, the "standard" tile is 21 pixels wide and 17 pixels high. I use Epyx's Art Director program for all my low-resolution color drawing, because its tools are the most flexible for this kind of work. Before doing anything else, I have to establish a working "grid."

On a blank screen of the background color, I draw an unfilled rectangle whose border is the width and height of a tile. Next, I cut out a copy of the rectangle (making it my paintbrush), pick a contrasting color from my palette (like blue if the first rectangle is orange), then switch my brush to a silhouette (whereby all non-background colors become the currently selected color). I stamp the now-recolored brush edge to edge with the original rectangle, making sure the horizontal top and bottom lines of each line up, and that they buttress each other, leaving no gap, but do not overlap (Figure 2a).

Next, I cut out both rectangles, switch the silhouette mode off (restoring the original colors), flip my new brush/block horizontally and paste it down right below the originals (Figure 2b). I now have a four-tile "checkerboard" of alternating colored rectangles. Finally, I cut out this whole block (very carefully, making sure I pick up no pixels beyond the edges of the rectangles) and fill the entire screen with it, making sure my fill point is the pixel in the upper left corner of the screen. When done, the screen is filled with a checkerboard of blank tiles (Figure 3). I use two different colors because it makes it easier to know exactly where one tile ends and the other begins.

(To do the above with DEGAS Elite you would create the first two rectangles manually [draw an orange one and a blue one next to it], then cut those, flip the block over and stamp it below the first two. Next you would clip all four, go to a blank workscreen and repeatedly stamp the block down to create the full-screen checkerboard.)

Usually there will be some cut-off rectangles at the right and bottom screen edges, which I usually erase.

Next comes the process of actually drawing the tiles. Usually I am given a list which specifies certain items that are "must haves" and sometimes where they must go on the grid (or grids if there is more than one screen of tiles). To start with I set up the color palette . . . or try to. It's usually impossible to throw together a final palette when you start, so I don't usually try. What I do is define those colors I know I will be using, and ignore the others.

For example, if the game has outdoor settings (like Paladin), I'll define two greens, two browns, etc.; if human figures appear, a flesh tone; blue for water, etc. Some games have color cycling effects (both Breach and Paladin use this), and I'll have to set aside two or three colors for this. I use the bare minimum number of colors I can get away with. So, a soldier with a green shirt will use the same greens used on trees, grass, etc.

Once I've defined a starting palette, I start drawing (usually terrain first, then objects, though, I rarely do all of one before starting on the other). I try to draw everything I can with the colors already defined, adding new colors only when absolutely necessary. This prevents me from too quickly running out of colors. I'll constantly be adjusting the values of the colors as I go, trying to make sure the hues work well wherever they must go (and constantly saving my work as I do this!).

As each tile is completed, I will make sure to eliminate all traces of the rectangular border which defines its edges (you must not draw only up to the border, but draw over it, because the lines of each rectangle define the outermost row or column of pixels in the tile). On terrain squares this is not usually a problem, as they tend to fill complete tiles, but on object squares, make sure to get rid of them once you no longer need them.

In the end, I will usually end up deciding to change all the pixels of one color to another on a selected series of tiles (often to free up a color register, or just to make things look better). This is easily done with *Art Director* because I can define a "window" around the specified tiles, and use "x-color" to replace one color with another or to swap two colors. (With DEGAS Elite you can replace colors by using "change.")

Drawing tips

The most common mistake with game graphics, and tile graphics in particular, is making things either too simple (and hence unappealing) or so detailed as to make the true nature of the item indistinct (even worse). In a square that might measure a mere one half by one half inches on your ST monitor, it's hard to get a good picture of almost anything. You usually have to spend a lot of time trying to make your terrain and objects really look like something!

In some cases, it's best to go with what I like to call "visual cliches," or items that are instantly recognizable. For *Universe III* I was required to create a pair of "UV Contacts" or ultraviolet contact lenses. Well, it was nearly impossible to make these lenses look like anything but little glass balls. In the end they were changed to glasses, because I could draw something that a player could identify at a glance and not be confused.

When drawing objects, I try to avoid having them touch the edges of their tiles, because then I can't make a mask big enough to overlap the size of the object. In some cases this can't be helped, but it's something you should pay attention to. The last thing you want is your on-screen persona's head to blend into a wall and his boots to seem to merge into the dirt at his feet.

Terrain squares that feature things like brick walls, wooden plank floors, grass, trees and water are not as easy to make as you might think. Trees and bushes end up looking like they're in an orchard if you draw one per tile. If your grass, dirt, etc., are too regular, it accentuates the fact that everything shown is laid out in tiles (you can't hide it, but there's no need to draw attention to it).

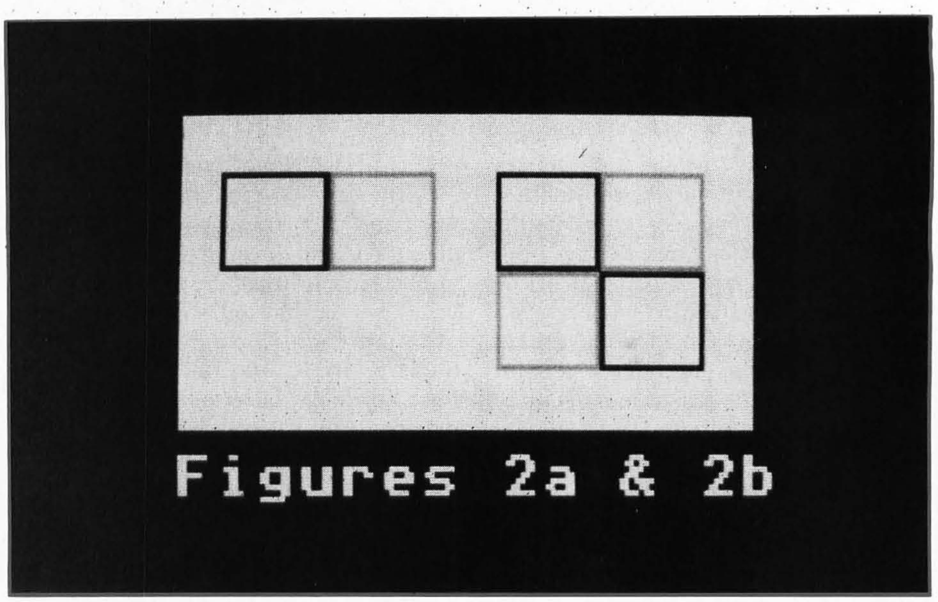
Try making such items overlap the tile edge, like having the left half of the tree against the right side of the tile and the right half against the left side. Water, bricks and such really must be designed so that when you line up a number of them they match up evenly, without obvious breaks. You have to fight to get this to look good, but it's worth it (Figure 4).

Making masks

As the object squares are completed, I'll start creating the masks. Most masks are merely one pixel larger around than the object, though in some special cases they'll be precisely the size of the object. Creating masks is simple, but the technique varies depending on the software you're using. With Art Director I merely fill in the mask squares with the specified color, clip the objects as brushes, make them a one-color silhouette, then stamp that onto the mask squares. Next, if I need the masks a pixel bigger than the objects, I'll set a "window" around the masks, and use Art's single-color "outline" mode (which automatically draws a single pixel wide line around a selected color).

With DEGAS Elite you would have to stamp the object down onto the mask square, eliminate all colors in the object, then manually outline it. Figure 5 displays a mask and how it can relate to an object.

When copying the objects to mask tiles, it's important to make certain that they are stamped down on the mask square precisely as they are related to their own



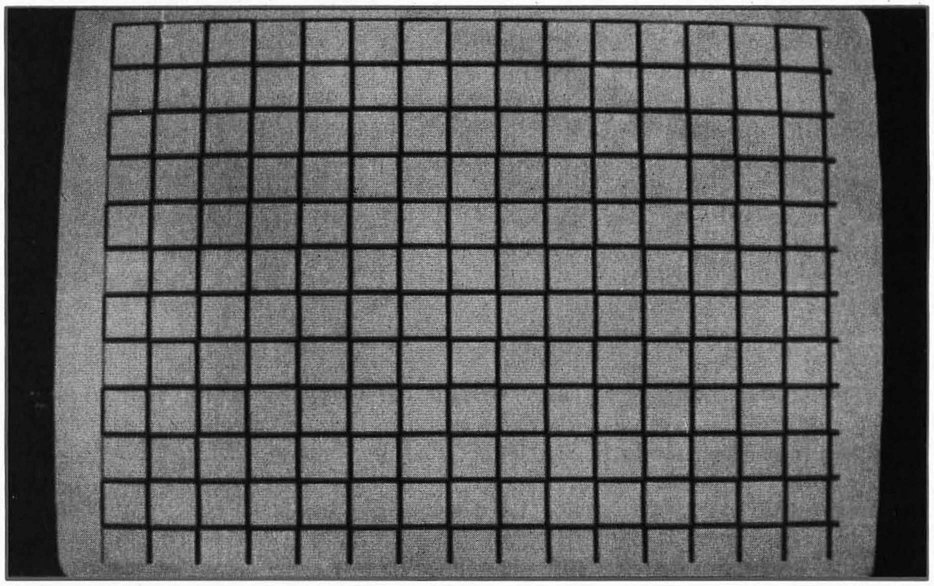


FIGURE 3

tiles. If they are even one pixel off, the mask will end up being in the wrong place, and the end result will not be pretty. Art has special keys which will allow your mouse cursor to move only horizontally (hold down Shift) or vertically (hold down Alternate) at one time. With DEGAS you may use the Alt-arrow and Alt-Shift-arrow key combinations to do the same.

Blits and sprites

Beyond tiles and vectors there's the usual game graphics which often consist of full-screen backgrounds and figures which move over that backdrop. The technique used to move the figures is the bit-block transfer or "blit," and the figures themselves are known as "software sprites." With this method, moving images are overlain upon a background plate. This is somewhat analogous to the object-over-terrain tile system, but more flexible because object sizes tend not to be so limited, and are generally free to move anywhere on the screen, rather than locked to a grid.

Games like *Joust* and *Star Raiders* fall into this category. Such programs often feature a lot of animation effects, such as flapping birds, banking aircraft, dynamic explosions, etc. How these are drawn and imported into the game depends on the programmer and his tools, but a lot of what I discussed earlier still applies. It's important to make sure your objects are distinct and recognizable, and not to use all your colors on one or two objects—or you won't have any colors left for the background, etc.

One of the most useful things is to import graphics from one program to another. For the Star Trek game demo I created, I built 3-D starships with Tom Hudson's *CAD-3D*, saved pictures of them at the required positions, then extensively touched them up (adding windows, running lights, torpedo tubes, etc.). Doing it this way sped up the process a lot and lent an air of realism to the animation. When a ship turned around on your viewer, the perspective and lighting remained correct and consistent.

Beyond this there comes the question of designing graphics for machines other than the ST. Admittedly, creating graphics for the Amiga on an ST isn't the best way to go, as you can't normally put enough colors on the screen to really approximate an Amiga display. (Those new 512 and 4096 color paint programs might change this, but we have to find a way to port those images to the destination machine.)

You can design IBM PC graphics (CGA and EGA mode) on your ST quite easily, because the screen resolutions are the same as ST low-resolution (320 x 200) mode. The only trick is to make sure to set up your color palette properly and make certain that either you or the programmers (if you are not one and the same) have the proper utilities for carrying out the conversions. It can be done—no doubt about it—but you have to play by the rules and understand the graphics limitations of the machine your graphics will be ported to.

I recently heard a horror story where a major software publisher hired an artist to create EGA graphics for a PC game and found the finished graphics unusable. The artist had created the graphics on an ST and used colors that had no counterparts in EGA. He'd gone into such detailing that recoloring didn't work. They had to hire another artist and do it from scratch!

And what about monochrome?

No, I haven't forgotten users of monochrome systems. Admittedly and unfortunately, most entertainment software on the ST is color-only. Graphics for monochrome have their advantages and problems. The benefits are in the fact that the higher resolution allows finer detail. The main drawback is that there are only two colors, black and white, which means the detail color allows isn't possible! Furthermore, you can't just replace one color with another, or stamp things over each other (it just doesn't work the same).

In place of color the artist must use outlines, dithered fill patterns (specific arrangements of black and white dots which create the effect of textures and gray tones). With tile graphics it's often best to try to leave any overlaid objects simple, without complex fills, and use the dithered effects on terrain squares. See Figure 6 for an example of the differences between the same tile in low resolution and monochrome.

Tools

The final question for the would-be graphics designer is "What software should I use?" There are as many answers to that question as there are possible techniques you can use in designing the graphics. Suffice to say the more graphics software you have, the better off you are. On any given project I may use Art & Film Director, DEGAS Elite, CAD-3D, and Cyber Paint, all in concert and with graphics tablets and video digitizers. Each tool has its strengths, and I am constantly porting graphics from format to format to take advantage of various tools.

A few software recommendations

Art & Film Director: If you primarily intend to work on low-resolution graphics, this two-in-one package provides the best graphics editing tools and special effects: up to 16 workscreens on a machine with a meg or more RAM, each with up to eight palettes. Graphics can be imported and exported in DEGAS .PI1 and NEO formats. Animation can be tested in *Film Director*.

DEGAS Elite: If you need to work in any and all ST resolutions, this is your best bet. Nice font handling and multiple workscreens (up to eight in one-meg or more RAM) are highlights. File formats are considered the ST "standard."

CAD-3D (get version 2.0 if you have one-meg or more RAM): If you want to simply create nicely shaded objects that can turn, this is what you need. Saves images in NEO and DEGAS format. You really must use this in conjunction with a paint program like those listed above.

Addenda

In "More Baudy Tales" (July 1988, ST-Log), at the very end of the next to the last paragraph of the section "Intro to the Hayes command set" (column 1, page 81) the line should read ". . .as in ATS0 = 0." Sorry if this little error caused any confusion.

When not writing articles for ST-Log, *Maurice Molyneaux continues to struggle with a nine-year-old science fiction novel, designs game graphics, consults for software companies and creates animated cartoon productions using microcomputers. Despite a ridiculously French name, he claims to have been born in Vicenza, Italy, and denies vicious rumors that he eats escargot and calamari while computing. He is the creator and director of the* Art and Film Director *sales video for Epyx—a ten-minute fully animated cartoon demonstration created entirely with ST systems. His DELPHI username is MAURICEM.*

School has started again, and it's the time of year where everything seems exciting and new: new teachers, new friends and even new schools for some people. This is the theme of this month's "ST User," as well; a new exciting product and plenty of news for the ST user. Take off those shoes, find a comfortable chair and read on.

ST USER

by Arthur Leyenberger

A major breakthrough

I just returned from the PC Expo held in New York City, and there was one product that blew me away. It was the Hewlett-Packard DeskJet printer. It gives "laser quality" output for the price of a high-end dot-matrix printer. If there ever was a printer for the rest of us, this baby is it.

The HP DeskJet uses inkjet technology that Hewlett-Packard pioneered a couple of years ago with their InkJet printer. That printer required specially coated paper, was very slow, and the output wasn't as good as the then currently available dot-matrix printers, such as the Epson FX-80.

"Dot matrix" simply means a matrix of dots that form the characters printed on a page. The method used to print the characters determines the quality of the output as well as the speed of the process and the cost of the product. Most home computer users usually use "dot matrix" to refer to the impact printers made by Epson, C. Itoh, Citizen, Gemini, etc. But the technology has come a long way since the early days of the Epson MX-70 with Graftrax.

Impact dot-matrix printers get the characters on the page by brute force. The matrix of dots is really composed of a number of pins that, when hit from the rear, poke the ribbon and put the ink on the page. The matrix of pins (the printer head) moves horizontally across the page while the page is advanced forward. To print darker characters on the paper, the printer head makes two passes per line and literally prints twice. To print the best near-letter-quality (NLQ) output, an impact printer may make up to three passes across the page and advance the paper in smaller increments than one line at a time.

Inkjet printers use liquid ink, much like a printing press, which is sprayed onto the paper through a matrix of microscopically small holes. The operation is familiar with the paper advancing past the horizontally moving printer head, but unlike impact printers, only one pass of the print head is necessary per line. If characters are to be made darker or bold, more ink is sprayed. If a fine hairline is needed, less ink is sprayed.

There are other differences too. The DeskJet uses individual sheets of paper of either letter, legal or European A-4 size. No more aligning the top edge of the continuous feed paper so that a multiple page document has the same centering at the top and bottom. As each page is printed, margins are exactly the same on each.

The DeskJet output is virtually indistinguishable from laser-printed output. The printer can support up to 300 dpi (dots per inch) text just like a good laser printer. But unlike a laser printer, it doesn't require multimegabytes of memory to do so. That's because the DeskJet prints the output in a real-time fashion, as it gets the data from the computer. A laser printer must first "map" the page in memory before it can print.

This is where the speed of the output differs. For one page, the DeskJet may be faster than a laser printer. But for multiple copies of the same page, once the laser printer has the image, it can spit out the additional pages one after another. The DeskJet requires the same amount of time to print each and every page. Draft mode is rated at 240 cps (characters per second) and letter quality is said to be 120 cps (about two pages per minute). The actual throughput speed will depend on the type of output (text or graphics) and the software that you are using.

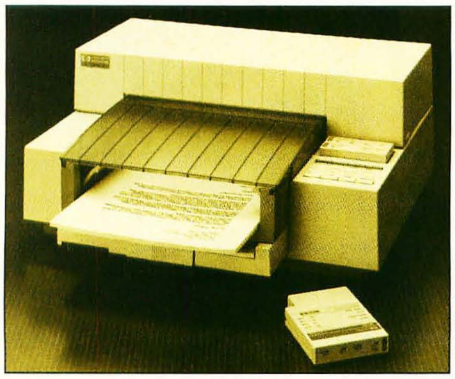
The HP DeskJet retails for $1,000 but I've already seen it advertised for about $600 to $700 at discount. It can Emulate an Epson printer by means of a $75 ROM cartridge, the ink cartridges cost about $20 a piece, and it has the support of Hewlett-Packard's one-year warranty and toll-free help lines.

There is only one minor flaw in the product as it currently stands. The ink requires several seconds to dry. In fact, the paper handling mechanism has a provision for not immediately dropping one printed page on top of another. As each new page is printed, the previous page is lowered onto the stack of output and the new one is held separately. The other problem is that, once the ink is dry, moisture will smear it, unlike a laser printer's output.

These are minor flaws, and the HP representative told me that they have chemists working on ink formulations "round the clock," trying to discover a non-smearable ink formula.

I hope to have a DeskJet printer soon. This is as revolutionary a product as the laser printer was when it first appeared. If you are in the market for a new printer, please check out the HP DeskJet.

The Hewlett-Packard DeskJet printer gives "laser quality" output for the price of a high-end dot-matrix printer.



All the news that fits

A lot is happening in the computer industry these days. Some of it applies to Atari directly while other news bears on Atari only indirectly. Most of the news, however, even though it does not apply to Atari directly, affects Atari users in one way or another. Here now, the news.

There is no question that Jack Tramiel is a difficult man to do business with. Many people dislike his rough and tough style of negotiation where Atari usually comes out ahead. But when the tables are turned, Atari is the first to scream "unfair playing field." Such was the case with the latest in technological snafus: the DRAM (Dynamic Random Access Memory) chip shortage.

Atari alleged that Micron, an Idaho-based chip manufacturer, agreed to sell some chips at an agreed-upon price and later reneged on the deal. Three million, 256 kilobyte chips were involved in the legal suit that Atari brought against Micron. Atari had sought damages for alleged violation of antitrust laws, breach of contract and bad faith. Sam Tramiel accused Micron of "destroying the competitiveness of the American microcomputer business" through its pricing practices. And Atari claims that the shortage of Mega STs in the U.S. is due to the chip shortfall.

The DRAM chips in question have been in increasingly short supply over the last year, due in part to the restriction on Japanese chip manufacturers. Because of these reduced imports, manufacturers of computers and video games have been unable to obtain enough chips to maintain full production. Micron is one of the few remaining U.S. manufacturers of DRAM chips and like other chip makers has been profiting from the shortage.

Recently, Atari announced that it reached an out-of-court settlement with Micron. Although Micron had filed to have the charges dismissed, the settlement was reached before the motion was brought before the court. Both Atari and Micron would not disclose the terms of the settlement other than to say that they each were "happy." Further, Atari is looking to acquire or start a chip manufacturing facility either in Asia or the United States.

A new WordPerfect?

In the recent hullabaloo concerning WordPerfect's supposed withdrawal from the ST software market, WordPerfect Corporation not only said that they would continue to market WordPerfect (WP) 4.1 for the ST, but that they would also be bringing out additional ST products, including a WP 5.0 version. The company would not disclose a ship date for the product (that got them into trouble once before), but they did say it probably would be out sometime in 1989.

I was able to view the PC version of WP 5.0, and to minimize the amount of drooling that is going on in the Atari ST camp waiting for WP 5.0, I'll mention some of the new features of that program. Of course, there is no guarantee that the ST version of WP 5.0 will look anything like or have the same features as the PC version.

The main differences between WP 4.2 and WP 5.0 for the PC and compatibles lay in better performance, improved documentation and new features. WP 5.0 has moved further towards being a desktop publishing program with increased use of multiple fonts, laser printer support and a multiple-view preview feature at two magnification levels. WP 5.0 has also improved its graphics handling features with the capability to import graphic file formats from *Lotus 1-2-3*, GEM, *Mac Paint*, *Autocad* and others.

I cannot possibly cover all of the details of the new PC version of the program in this space. But I can say that with this new release, WordPerfect has maintained its position as the best PC word processor currently available. For ST users, WP is not only the best word processor, its really the only one in its class. It will be interesting to see what WordPerfect Corporation does to the ST version to make it even better.

Hand in hand

Do you remember the early days of Atari computing when you could buy an Atari computer at many department stores such as Sears or Macy's? At the time you could probably buy a Commodore machine at even more places, but that is a topic that has been previously discussed. At any rate, Atari and Commodore dominated the market.

I remember my local Sears store (hey, it's where America shops, right?) had Atari XL, Commodore 64, Apple II and Coleco Adam (huh?) computers all lined up on several display aisles. Software and hardware were sold by clerks who didn't know a disk drive from a toaster, but the stuff was being moved in *big* quantities. Then the bottom fell out of the market. Real PCs (I'll get some hate mail here) became less expensive and more common too. I don't think that same Sears store has had a computer in stock in years. Of course, there was and still is the Sears Business Center that caters to IBM PC and clone machines, but that is a separate store.

The tide has turned again. Department stores are bringing back PCs, but the difference is that now those PCs aren't Atari PCs. Catering to the home business user and families, department stores such as ZCMI, Macy's, Lazarus, Sears, Burdines and Dillards are stocking Apple, Epson, Amstrad, AT&T, Commodore and Blue Chip computers selling from $500 to $2,000. In a recent survey of the above-mentioned stores, only one (Utah-based) ZCMI was carrying an Atari product: the 520ST and 1040ST.

The department stores are targeting unsophisticated users who want to trade up to a low-end system from a typewriter or word processor. The stores are also packaging systems into applications such as word processing in order to draw customers in and convince first-time users to buy. A bundled system makes it easier for

(to page 84)

STAY ON THE CUTTING EDGE!

12 Issues Only $28

Save $14 Off The Cover Price!

12 Issues *With Disk* Only $79

New Lower Price!

SUBSCRIBE TO ST-LOG AND SAVE!

The world of ATARI-ST continues to grow by leaps and bounds and ST-LOG is there every step of the way! SUBSCRIBE and SAVE today!

☐ 12 Issues $28 (MCJWW) ☐ 12 Issues with Disk $79 (DCJWW)

☐ PAYMENT ENCLOSED ☐ BILL ME

CHARGE MY: ☐ VISA ☐ MC #__________

EXPIRATION DATE SIGNATURE

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615.

Offer expires December 31, 1988.

ST LOG
THE ATARI ST MONTHLY MAGAZINE

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**

● ● ● ● ● ● ● ● ● ● ● ● ●

WATCHCART ST

The worlds most inexpensive clock cartridge. Finally its affordable to keep your time and date accurate. 3 year battery included. **ONLY $24.95**

MEGADISK Ultra high speed solid state disk drive ● 500% Faster than a Hard Disk ● Provides almost instant booting ● Like a RAM disk that's always loaded with your favorite programs and ready to use ● One megabyte of Solid State storage ● Built in battery back-up in case of power failures

MEGADISK is actually one megabyte of RAM that simply plugs into your cartridge port. It acts as an added disk drive that's ultra fast and always ready for use. Like a Hard disk, MEGADISK won't loose its memory when your computer is turned off. It comes with its own power supply and battery back-up system so its independent of your computer.

Megadisk can be configured according to your needs. ● Set it up as one large disk ● An 800K double sided disk and a 200K hardware print buffer ● Or as two 400K single sided disks and a print buffer

Megadisk will work fine with your current system whether you have a hard disk and two drives or you're just getting started.

Megadisk is perfect for those who want the high speed of a hard disk for a lower price. Its even better for power users or software developers who may already own a hard disk and two drives but want extra speed and power. Megadisk can also emulate other cartridges for testing and back-up. In addition Megadisk can be used with Switch/Back to allow you to instantly jump between two full size one meg applications. **$299.95***

Price Subject to change

Megadisk Clock Option – Adds a Clock/calendar card to your Megadisk cartridge. Contains replaceable Three year battery 29.95

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) **Only $199.95**
(Clock option card is also available for Polydisk $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.

Power Print 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**

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.
DIGISOUND 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**

DIGIPLAYER The High powered digisound software can now be obtained by those who already own a digitizer for the ST. Compatible all cartridge based digitizers. Extend the power of your digitizer with Digiplayer.

Only $49.95

ALPHA SYSTEMS

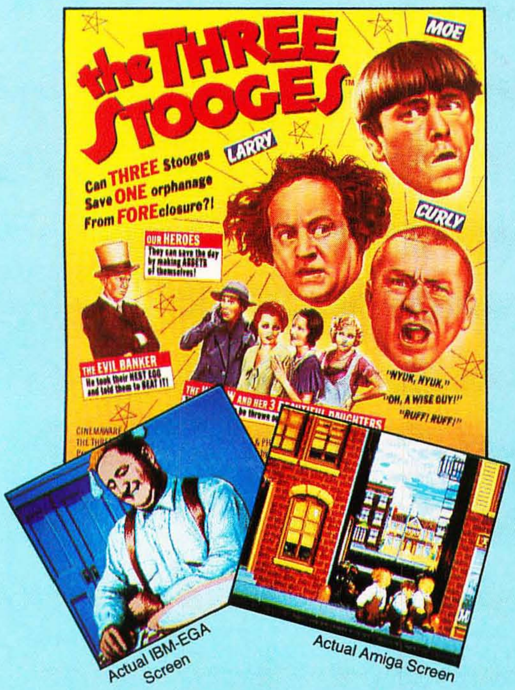
24 HOUR HOTLINE – VISA & MasterCard Welcome
216-374-7469
Customer Service line (216) 467-5665. Call or write for free catalog.

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 #125 ON READER SERVICE CARD.

CINEMAWARE

PRESENTS


the THREE STOOGES
MOE
LARRY
CURLY
Can THREE Stooges Save ONE orphanage From FOREclosure?!
OUR HEROES They can save the day by making ASSETS of themselves!
THE EVIL BANKER He took their NEST EGG and told them to BEAT IT!
"NYUK, NYUK, NYUK."
"OH, A WISE GUY!"
"RUFF! RUFF!"

Actual IBM-EGA Screen

Actual Amiga Screen

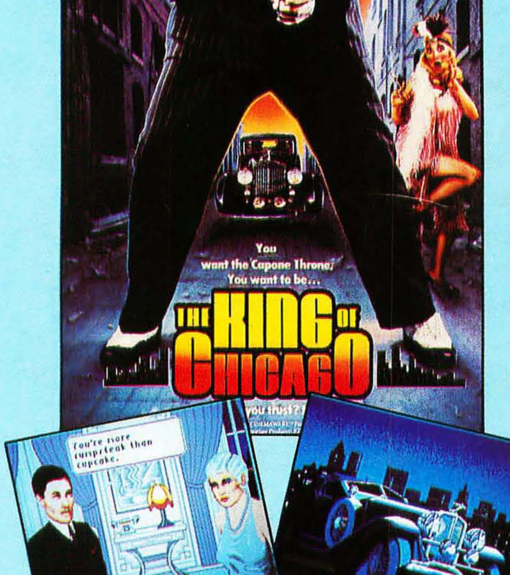
Actual Apple IIgs Screen

Actual C64 Screen

ROCKET RANGER
Only YOU can stop the Interplanetary Spread of Jet-Propelled NAZI FASCISM!
SEE THE BEAUTIFUL YOUNG DAUGHTER OF A FAMOUS SCIENTIST HELD CAPTIVE BY A GANG OF GODLESS GOOSESTEPPERS!
A Master Race from Outer Space threatens the Fate of the Free World!

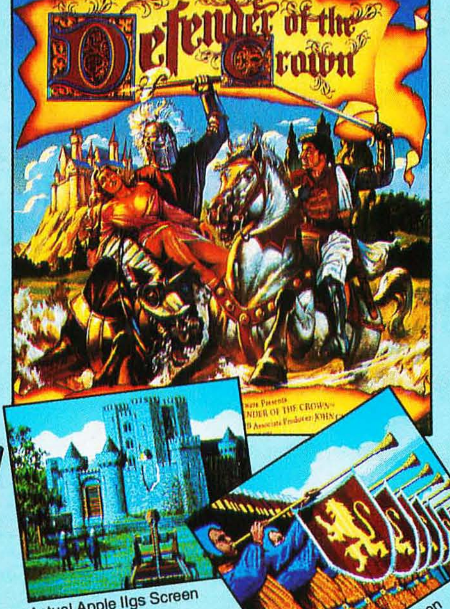
Actual Amiga Screen

Actual C64 Screen


You want the Capone Throne, You want to be...
THE KING OF CHICAGO
You're more cumpstank than cupcake.

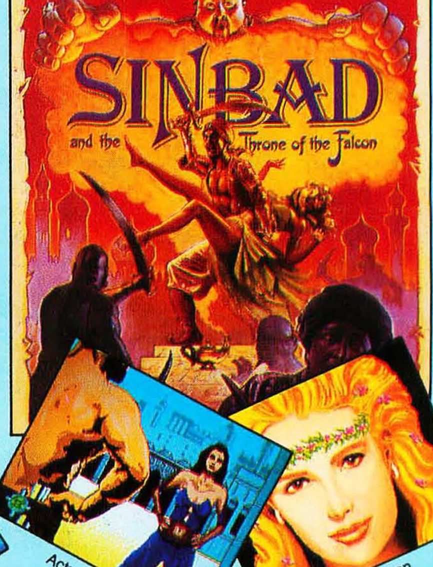
Actual Amiga Screen

Actual Atari ST Screen



Actual Apple IIgs Screen

Actual IBM-EGA Screen



Actual IBM-EGA Screen

Actual Atari ST Screen

NOW PLAYING AT A SOFTWARE DEALER NEAR YOU

Cinemaware Corporation, 4165 Thousand Oaks Blvd., Westlake Village, Ca. 91362 Tel: (805) 495-6515
Available for Amiga, Atari ST, Apple IIgs, IBM PC and Commodore 64, which are trademarks respectively of Commodore-Amiga, Atari Inc., Apple Computer, Inc., International Business Machines and Commodore Electronics, Ltd. Not all products are available for all formats.

CIRCLE #126 ON READER SERVICE CARD.