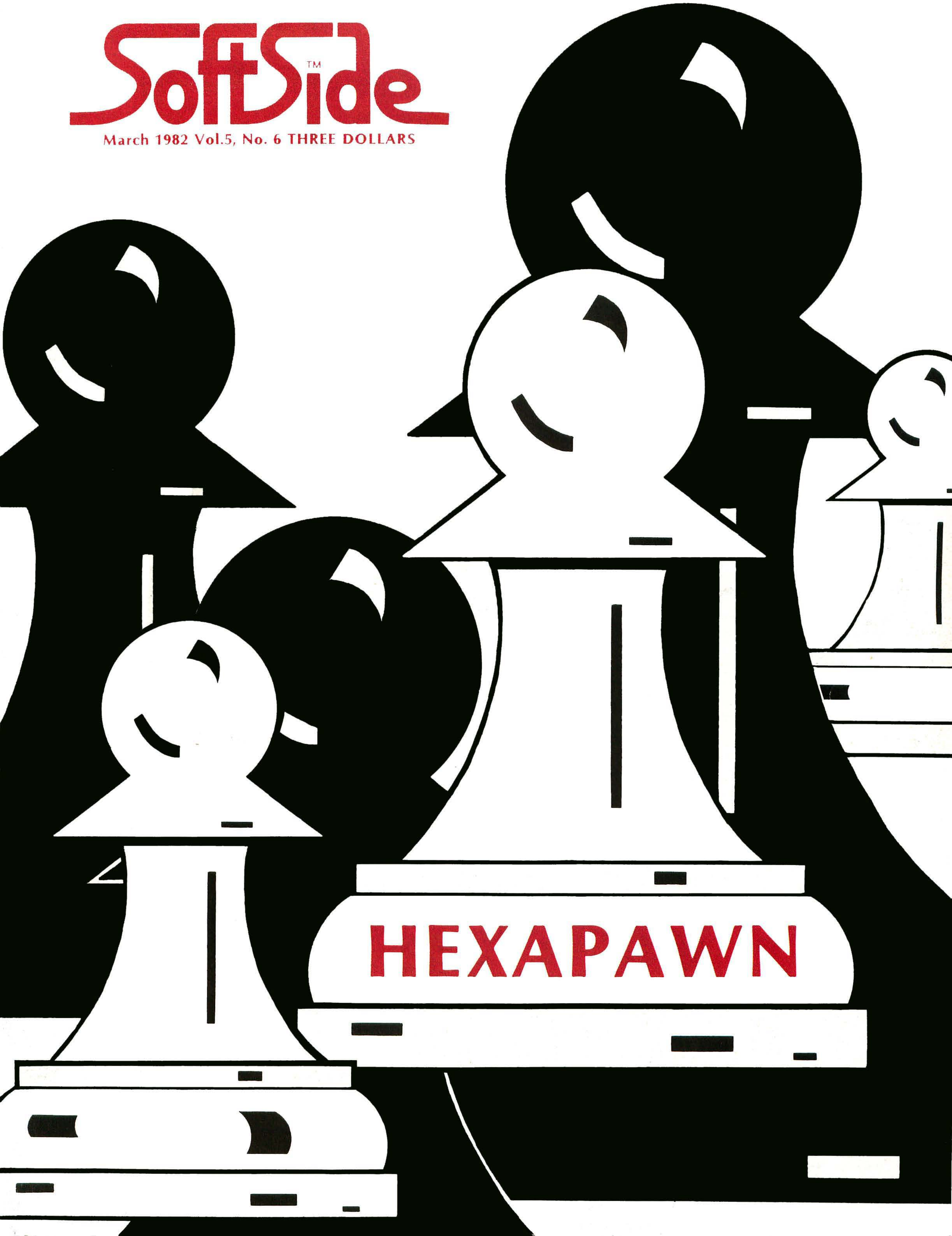
SoftSide™

March 1982 Vol.5, No. 6 THREE DOLLARS



STOP TYPING!

Get Instant Enjoyment from SoftSide's programs with SoftSide's Cassette Version (CV) and Disk Version (DV)!

Our media editions let you spend less time TYPING — and more time USING the fine software that **SoftSide** brings you every month. And we let you choose the version you want.

Cassette Version (CV)

SoftSide's Cassette Version (CV) offers you an inexpensive way to enjoy our programs without hours of typing or hunting for errors. All programs are tested and ready to go!

CV gives you the programs offered for your system each month in **SoftSide** on a tape, plus the magazine itself — 12 magazines and 12 tapes per year for just $75.

Disk Version (DV)

DV contains a BONUS program for your system on the disk in addition to the other programs available that month. Only the documentation for the bonus programs will appear in **SoftSide** magazine, NOT the code. The bonus programs will be of every conceivable type — multiple and Machine Language programs, modified languages, ongoing modular programs and software so extensive, it would take an entire issue of **SoftSide** just to print the code.

Feel like you're missing something? You are. Don't wait to take advantage of our offer — 12 magazines and 12 disks for just $125 a year. For orders outside the U.S., add $50. For your convenience we also offer an installment payment plan for MasterCard and VISA holders: Pay just $32.50 per quarter (a total of $130 which includes a $5 billing charge).

To order, use the card provided in this issue.

A feast of computing ideas.

If you work with a 6502/6809-based system, you're probably hungry for the facts and ideas that will help you understand the inner workings of your computer. You want to go beyond canned software—use your computer for more than games—learn the advanced programming techniques that enable you to get the most out of your 6502/6809 system.

MICRO, The 6502/6809 Journal, gives you page after page, month after month, of solid information to sink your teeth into. **MICRO** is the premier how-to magazine for serious users of the Apple, PET/CBM, OSI, Atari, AIM, SYM, KIM, and all 6809 based systems including the TRS-80 Color Computer. It's a resource journal internationally respected by professionals in business, industry, and education. Every issue of **MICRO** keeps you informed with up-to-the-minute data on new products and publications:

- **hardware catalog** with organized, concise description
- **software catalog** in an easy-to-use format
- **new publications** listed and annotated
- **reviews and evaluations** of significant products

And there's much more:

- **In-depth hardware tutorials** bring expert advice into your home or office.
- **Detailed discussions of programming languages** deepen and broaden your programming ability.
- ***Complete program listings*** enable you to increase your machine's capabilities.
- **Bibliography of 6502/6809 information** helps you to find pertinent articles in a timely manner.
- **Special monthly features** with in-depth treatment of one subject or system increase your knowledge of the field.
- **Balanced mix of machine-specific and general articles** for your everyday use as well as long-range reference needs.
- **Informative advertising** focused specifically on 6502/6809 machines keeps you abreast of latest developments.
- **Reader feedback** puts you in touch with other micro-computerists.

MICRO is the magazine you need to get the most from your own 6502/6809 system!

To order, send your check or international money order (payable to MICRO) and the order form at left, to:

Subscription Fulfillment
MICRO, Dept. MI
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Or, for your convenience, call our toll-free number:

1-800-227-1617, Ext. 564
(In California, 800-772-3545, Ext. 564)

and charge your subscription to your MasterCard or VISA. (All orders must be prepaid in U.S. dollars or charged to your MasterCard or VISA.)

SUBSCRIPTION RATES (U.S. dollars)
Yearly subscription (ISSN 027-9002) saves 20% off the single-issue price.

U.S. $24*
Canada $27
Europe $27 ($42 by air mail)
Mexico, Central America, Mideast, North and Central Africa $27 ($48 air)
South America, Far East, South Africa, Australasia $27 ($72 air)

* SPECIAL OFFER—U.S. ONLY: Save even more—30% off single-issue price: 2 years, $42

Dept. S S

NO. 41
$2.50
OCTOBER 1981
MICRO
THE 6502/6809 JOURNAL
Special PET/VIC Feature
A Look at the TRS-80 Color Computer
Step Up to Programmable Motion
A to D Converters for Your Computer

You'll love every byte.

YES! I want to get more from my microcomputer. Please send me

___ year(s) of MICRO at $______/year.
(Outside U.S. and Canada, please indicate via ☐ surface or ☐ air mail.)

Name ____________________

Company ____________________

Street ____________________

City ________ State ________ Zip Code ________

☐ Check enclosed for $ ________
☐ Charge my credit card account
☐ VISA ☐ MasterCard

Signature ____________________

Card number ____________________

Expiration date ____________________

SoftSide™

PUBLISHER
Roger W. Robitaille, Sr.

ASSOCIATE PUBLISHER
EDITOR
Randal L. Kottwitz

PROGRAMMING EDITOR
Jon Voskuil

EDITORIAL DEPARTMENT
Scott Adams
Rich Bouchard
Mary Locke
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Alan J. Zett

PROMOTION ART DIRECTOR
PROMOTION MANAGER
Nancy Lapointe

ADVERTISING MANAGER
Clif Campbell

PRODUCTION MANAGER
Lynn Wood

ASSISTANT TO THE EDITOR
Dave Albert

PRODUCTION DEPARTMENT
Lynda Fedas
Karen Lawrence
Tom Stanton

CUSTOMER SERVICE
Jean Matthews

DEALER SALES
Kathie Maloof

STAFF
Brian Berkebile
Jerry Butler
Mary Edwards
Donna Jean
Bea Kimball
Doris Miller
Cindy Schalk

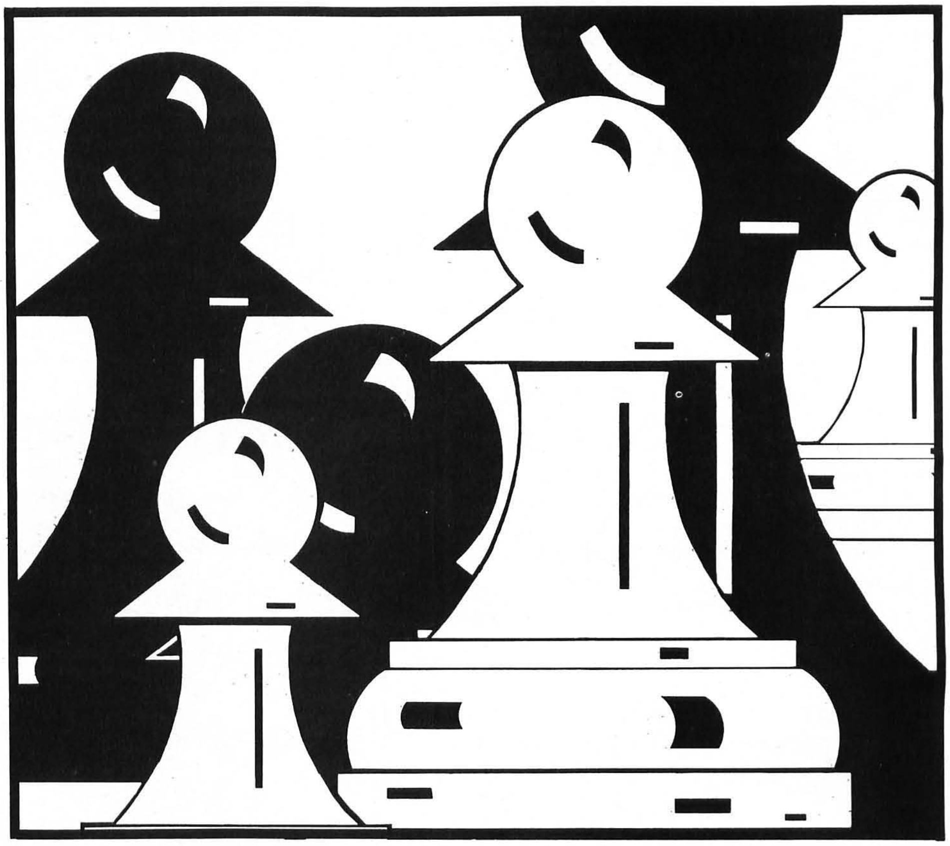
Photographs by Dean F. H. Macy

SoftSide is published each month by *SoftSide* Publications, 6 South Street, Milford, New Hampshire 03055. Telephone 603-673-0585. Second class postage paid Milford, New Hampshire and pending at additional mailing offices. ISSN: 0274-8630. Subscription rates: USA and Canada, $30.00/12 issues. USA and First Class APO, FPO, Mexico, $40.00/12 issues. Overseas air mail: $62.00/12 issues. Media subscription rates: Magazine and cassette, $75.00/12 months. Magazine and disk, $125.00/12 months. APO, FPO, Canada, Mexico, (add), $20.00/12 months, other foreign (add) $50.00/12 months.. All remittances must be in U.S. funds. Mail subscription inquiries to *SoftSide* Publications, 515 Abbot Dr. Broomall PA 19008. Entire contents © copyright February, 1982, *SoftSide* Publications. All rights reserved.

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

If you do not receive your April issue of *SoftSide* by April 6, contact *SoftSide* Publications, 515 Abbot Drive, Broomall, PA, 19008 or call 1-800-345-8112 (In PA call 1-800-662-2444).


Cover illustration by Lynn Wood and Nancy Lapointe

CONTENTS

DEPARTMENTS

APPLE™/SIDE

ATARI®/SIDE

TRS-80®/SIDE

INTRODUCING PERFECT WRITER.™ NOW, GET THE FINEST WORD PROCESSOR AT ANY PRICE FOR JUST $289.00

AND GET OUR INCREDIBLE, MONEY-BACK GUARANTEE!

That's right. If you own an Apple II, Heath/Zenith, TRS-80 model II, North Star, Osborne 1, Superbrain, Micropolis/Vector, or CP/M 8," you can now afford the most advanced, yet easy-to-use, word processor on the market today.

But don't take our word for it. See for yourself.

We're so sure you'll be amazed by Perfect Writer's easy-to-use features and capabilities, that we'll let you try it for thirty days and give you your money back if you're not completely satisfied!

FEATURES AND FLEXIBILITY YOU HAVE TO SEE TO BELIEVE.

Looking for the advanced features you'd expect to find in the most expensive word processors? Perfect Writer has them. Like automatic formatting of letters, out-lines, reports, memoranda, articles, books, or other types of documents.

What makes Perfect Writer unique, are the features you won't find in word processors costing many times its price.

Like the incredible sophistication of Virtual Memory Architecture, that lets you edit a full-length novel as easily as you'd edit a letter.

Or a remarkable breakthrough called Split Screen Editing™ that lets you divide the screen into two parts, each showing a different document, and transfer sections of text between the two!

Perfect Writer. The remarkable word processor combining sophistication and simplicity.

And that's just the beginning. Perfect Writer offers much, much more:

- Automatic numbering of pages, chapters, sections, footnotes, internal page references, and automatic generation of table of contents and indexes.
- Mail merging capability allows generation of form letters and more.
- Automatic carriage return, auto-centering, tab setting, and automatic placement of footnotes.
- Quick and easy cursor commands.
- Easy deleting and inserting procedures.
- Safe and convenient storage of materials.
- Searches for and replaces words and phrases in a document, including near misses.
- Full range of printing options, including right margin justification, micro-spacing, super and subscripts, italics, and bold face.
- And lots more.

MONEY BACK GUARANTEE.

Try Perfect Writer for 30 days at our risk. If you're not completely satisfied, return it for a prompt, complete refund.

PERFECT WRITER IS EASY TO USE. WE GUARANTEE IT.

When we say that Perfect Writer is easy to use, we mean it. That's why we're not afraid to offer a money back guarantee.

You see, Perfect Writer was designed to be simple. In fact, you don't even have to read our crystal-clear manual in order to use Perfect Writer. The pictures, diagrams and illustrations tell you all you need to know, so you're off and running in no time.

There are no difficult and awkward procedures and commands to learn. We guarantee it.

ORDER NOW. WHAT HAVE YOU GOT TO LOSE?

If you'd like to own the most sophisticated word processor on the market today at the current, introductory price of just $289.00, we invite you to order now.

Perfect Writer is completely guaranteed, so there's no risk of any kind. Why not see for yourself what this incredible new software breakthrough can do for you?

TO ORDER CALL TOLL FREE: 1-800-221-2486
IN NEW YORK CALL: 1-212-619-4000

WRITE: COMPUTER SERVICES CORPORATION OF AMERICA
71 MURRAY STREET
NEW YORK, NEW YORK 10007

Yes. I want to see for myself if Perfect Writer is as great as you say it is. So please send it to me, subject to the guarantee printed on this page.

☐ Perfect Writer and Manual $289.00

I also want to order, subject to the same guarantee:

☐ Perfect Speller $189.00 ☐ Perfect Sort $189.00
☐ Perfect Mailer $189.00

Name ______________________

Address ______________________

City, State, Zip ______________________

☐ MasterCard ☐ American Express
☐ Visa

Number ______________________ Expires __________

TRY THESE OTHER PERFECT SOFTWARE PRODUCTS
(THE SAME MONEY BACK GUARANTEE APPLIES).

Perfect Speller™: The best automatic proofreader available! Perfect Speller uses its 20,000 word dictionary to proofread your document and catch all your spelling mistakes. (Dictionary size limited only by your system's storage capacity.) You can add words to the dictionary, build separate dictionaries, change the dictionary, and more. $189.00 guaranteed.

Perfect Mailer™: It's more than a mail merging or form maintenance system! Perfect Mailer can organize and sort your correspondence, develop sophisticated mailing systems, and keep track of your correspondence. Don't miss this remarkable new software innovation! $189.00 guaranteed.

Perfect Sort™: Here's a sort/merge/extract program that operates on all kinds of data! Records can be fixed or variable, with a variety of variable length fields. Files can be examined using comparison and logical operators. Want speed and efficiency? You've got to have Perfect Sort! $189.00 guaranteed.

EDITORIAL

Morning Shower

By Jon Voskuil

I do my best thinking in the shower. Which means that my day peaks between 6:35 and 6:45 a.m., and it's all downhill from there. What I need, I've been telling myself, is a double-insulated, waterproof, portable computer that I can take into the invigorating spray with me. Imagine the programs that might flow forth from that few minutes' encounter each day, instead of getting washed down the drain with the shampoo lather. (I can NEVER remember my brilliant showertime insights long enough to get them down on paper or disk afterwards.)

Alternatively, I could try fitting my Apple with a voice-recognition input device, which would mean that I'd only need a steam-proof microphone to suspend from the ceiling overhead. The trouble with that (aside from stifling my operatic inclinations) is knowing how the heck to pronounce all those weird BASIC and DOS commands. I mean, learning how to read and write computer talk is one thing, but speaking it is something else. I've never even been able to get myself to dictate in English, let alone BASIC. How do you pronounce FP? Or how about STR$(? My tongue isn't made to say more than one or two consonants in a row, especially when they're followed by two non-alphanumeric characters like $ and (.

The ultimate solution, I suppose, is human-machine symbiosis. That's what author Michael Crichton wrote about some years back in his book, *The Terminal Man*. It seems his hero was having seizures, so some innovative scientists wired his brain to a computer via an implanted radio transceiver. The computer could detect the onset of a seizure through electrodes in the guy's brain, and could then send an electronic counter-impulse to nip it in the bud. Of course the symbiotic relationship, cozy as it was, eventually went haywire and caused everybody lots of grief. But that's the way it always goes in good suspense novels.

Just imagine the potential of direct-wiring to a computer. How easy it would be to program computers, if all you had to do were to think the program into existence! That sounds a whole lot faster than typing. The disadvantage, on the other hand, would be having all those computer error messages routed directly into your cerebral cortex. When I think of how annoying it is at times to have my Apple beeping and messing up the screen with "SYNTAX ERROR" and "FILE NOT FOUND", I wonder what it would be like to have the same beeps and messages delivered through a pair of electrodes. At least I wouldn't be embarrassed by having other people overhear all my errors. Unless the thing started taking over my vocal cords, that is. (Wouldn't THAT add new meaning to the term "Adam's Apple"??)

Think of the possibilities for the *SoftSide* of the Future, if a significant number of our subscribers were direct-wired to their computers. No more mailing bulky paper magazines; no more worrying about disks getting shipped next to one of those big, hairy magnets from Edmund Scientific. All we'd have to do is give each subscriber an access code, and set up a toll-free phone number for you to call, and we'd download *SoftSide* into your ear in about 20 seconds. We could cut subscription rates, eliminate all our media duplication problems, and give higher payments to our authors (and editors).

Of course, this type of symbiosis would dramatically affect the computer programs of the future. Imagine playing a three-dimensional graphics adventure with four-voice stereo sound and speech synthesis, inside your brain. And would you believe a symbiotic version of *Space Invaders*? Or *VisiCalc*? How about a computer-enhanced multi-player fantasy role-playing game? I'm going out tomorrow morning to get a trademark on the name of my future blue-chip company: Symbiotic Software.

Hey! Who took my towel?!. . .

K-Byters

ANOTHER PROGRAMMING CHALLENGE

Some time ago ***SoftSide*** began inviting its readers to submit "One Liners" — self-contained single-line programs for the TRS-80®, Apple, or ATARI® which would provide a continuously changing graphics display. The response has been excellent, and we're still looking for more submissions.

Now we have a new challenge for you as well: "K-Byters." A K-Byter is a BASIC program which fits into 1K (1024) bytes of program memory. There aren't any restrictions on the nature of the program, other than its size. It can be a graphics display, a game, a mini-adventure, or anything your imagination and programming skills can create.

Note that the program does not have to RUN in 1K of memory; it can use as much RAM for arrays, strings, graphics mapping, etc., as you need. We'd prefer that it be able to run in a 16K system, but this not an absolute limit.

Here then are the official rules:

1. The program must be written for the Apple, TRS-80®, or ATARI® entirely in BASIC (although it may create and call Machine Language routines).
2. The program must occupy no more than 1024 bytes of memory before running.
3. The program must be submitted on tape or disk, accompanied by your name, address, phone number, and a brief written description of its operation.
4. The tape or disk will be returned only if accompanied by a self-addressed envelope with adequate postage AFFIXED (do not send money).
5. Winners will have their programs published in ***SoftSide*** and will receive a $10 software certificate for their programming excellence!

Send submissions to:

K-Byters, c/o ***SoftSide***
6 South Street
Milford, NH 03055

From our readers

GAMBLER

Dear *SoftSide,*
Randy Hawkins should be complimented for his excellent program *Gambler* that ran without a hitch. I especially liked the sound additives that enhanced the program. The sound routines were simple and most programs should contain sound because of the interest it adds.

Joseph F. Dineen
Pocasset, MA

DOSPLUS

Dear *SoftSide,*
I have been a subscriber since the inception of your magazine. I liked the small *TV Guide* format dedicated to the TRS-80®. Slowly I have become adjusted to the "new" *SoftSide.* The cassette version was done well and now the DV is superb.
A few thoughts have come to me about the magazine and its subscribers. First: It seems you cover a very wide range, from the 16K Model I up to the 48K "super" Apple. Second: You seem to have a good mixture of magazine, cassette, and DV subscribers. Now do us disk users a favor. When you write the articles about random and sequential files give us a break. Write the articles as if we knew nothing and build up. Don't duplicate Tandy's uninformative *Disk Users Manual.*
Also, how about giving us DV users more information about DOSPLUS and T-BASIC? The way you set up NEWBASIC was terrific. How about the same for DOSPLUS and T-BASIC?
Keep the bugs out and good bytes coming.

Anthony J. Asaro
Howell, NJ

Editor's Reply: In February we began a series of informative articles on DOSPLUS. We will continue to run the series until we have exhausted the available material.

THE *SOFTSIDE* SAMPLER

Dear *SoftSide,*

I have been a reader of your magazine since the first Apple edition. I have a suggestion for your incredible magazine. Why don't you let all those people who don't get your magazine benefit from your programs? You could put out a book containing all your programs, K-Byters, and One Liners in separate sections according to each computer. It would also be convenient for all us subscribers to have the programs at our fingertips!
How about it?

Matt Laurence
Lincoln, MA

Editor's Reply: Your wish is answered. In January, Hayden Book Company, Inc. in cooperation with *SoftSide* Publications released a book entitled *The SoftSide Sampler, TRS-80® Entertainment Programs.* Although this book has only selected TRS-80® programs, we have plans for future releases for other systems and other types of software.

COMPUTERS BY MAIL

Dear *SoftSide,*
I could just scream! Last week I called a reputable mail-order house to order a 48K Apple system with disk drive, etc. The answer I got to the order request was that they were sorry, but the Apple Computer Company no longer allowed their dealers to sell by mail order! I couldn't believe it! So I decided to call all of the toll-free (I'm always looking for the cheapest way to do things) computer houses that I could get numbers on. Alas, they all refused to sell to me for fear of losing their dealerships. It seems that Apple is trying to force people to go to the local store and settle for less than the best deal! This is an outrage! I have saved for over a year in order to buy a complete system at a price that I can afford, and now Apple is telling me that I have to come up with the extra bread so I can support a greedy chain-store operation that has this area sewn up! Could it be that IBM is giving Apple the heebie-jeebies?
Well, rest easy Apple, IBM won't get my bucks — they already get enough from the use of the mainframe at my employer's offices. But don't think that Apple is going to get my dough either. I'm going for the ATARI® 800 system, which is less expensive, and, thanks to Apple forcing me to investigate deeper, I now believe the 800 has far greater potential for software development.
So, fellow hackers, I ask that you support free enterprise by letting Apple know about your disgust. They will probably come up with some great excuses, but those explanations won't help you or myself in the wallet. And while you're at it — buy an ATARI® ! (That's purely selfish — I want to see more ATARI® programs in *SoftSide!*)

Jack McKirgan II
Washington C.H., OH

Editor's Reply: Although I can sympathize with your frustration, I must admit that when a friend asks me where to buy a computer system, I send him to the nearest retailer. There's no doubt that you pay a premium at any retail outlet. However, much of that premium pays for "service after the sale." The more complicated any piece of electronic equipment, the more malfunctions become possible. When the inevitable day arrives that your system needs repair, you'll be much better off delivering it to the shop yourself rather than entrusting it to UPS or (God forbid) the U.S. mail.

GEAP

Dear *SoftSide,*
Thank you for your review of my *GEAP* (*Graphics Editor and Programmer*) program. Readers might like to know that they can order *GEAP* from:
JF Consulting
74355 Buttonwood
Palm Desert, CA 92260
Phone (714)340-5471
In her review, Ms Grothman mentioned that she sometimes accidentally hit the spacebar, losing the picture on the screen. In the most recent versions of *GEAP* (versions 1.3 and 2.0), we require pressing two keys for "dangerous" commands, hopefully preventing such accidents.
Thanks again for the review.

Bill Mason
Hornitos, CA

MUSIC IN THE MICRO REVISITED

Dear *SoftSide,*
Usually I cringe when I see an article comparing our products with other products. Usually the author has little time to learn the details of the two (or more) products and it shows, more often than not. But the review of our nine-voice music card and Mountain Computer's *Music-System* in the November, 1981, issue seems to be one of the exceptions. The author, Christopher U. Light, has obviously spent some time getting to know both systems well, and he's really interested in using the systems, too.
I've read other comparisons, and have always wished that someone would pick a simple song and enter it on each system being reviewed; finally Mr. Light has answered my wish. Naturally, I'm pleased to hear that Bach's *Invention Number One* took him only an hour and a half with our system and about six hours on the Mountain Computer system; we've put a lot of

work into making our product easy to use. I wonder if Mr. Light would try the test again a year from now when he's really used to both systems and see how much faster he is with each system; that would really be interesting.

I was also pleased to read that he had noticed the "human engineering" factors we put in, such as "clicks" during note selection (he even mentioned that this allows you to enter music without looking at the screen as much, which is exactly what we had in mind), the very fast response to virtually all commands, and also the automatic measure bars and accidental continuation. Most reviewers have failed to notice any of these subtle touches.

Much as I admire the effort Mr. Light has put into the review, I can't help but point out some minor rough spots. In particular, our music card is not called the "Alf II"; but I can hardly fault Mr. Light since there has been some confusion over names. The card was originally called the "Apple Music II," but Apple Computer Inc. objected to use of the word "Apple" (although they knew about the name for quite some time without any comment). We changed the name to *Music Card MC1*, and also changed the name of our first music card for the Apple (referred to as the "Alf I" in the article) from "Apple Music Synthesizer" to *Music Card MC16*. Obviously, the "MC" of each model number stands for Music Card; we didn't realize at the time it was also Mountain Computer's initials (they had only recently changed their name from Mountain Hardware) but I don't think there's been any confusion. The list prices are slightly lower than stated in the article: $195 for the *MC1* and $245 for the *MC16*.

I'd also like to correct Mr. Light's statement that neither system has quintuplets; he's confusing what our system will *display* and what it will *play*. True, our system doesn't display quintuplets or notes shorter than sixty-fourth notes in standard notation, but it does play a variety of short and non-standard durations. They appear as "x" notes on the screen, and in fact the example for x-notes in the manual is a set of five quintuplets. The Mountain Computer system is limited to thirty-second notes and doesn't have quintuplets (or even triplets), as mentioned in the article.

I would like to add that while ALF doesn't supply a music merger program with the card like, as Mr. Light points out, Mountain Computer does, we do offer one separately in a package of music entry/playback aids. However, Mr. Light does point out in the article that *Invention* would have fit in memory over 11 times with our system and only one-third of it fit on the Mountain Computer system; so you can see a merger program is rarely needed on our system. Also, the "5396 free bytes" after entering *Invention* is really 5396 free *notes*; that's 16,188 bytes.

Once again I would like to compliment Christopher Light — and *SoftSide* — on an excellent review.

Philip Tubb
ALF Products, Inc.
Denver, CO

Author's Reply: ALF's separate package, which wasn't available when I bought my synthesizer, not only contains a merger program but a number of other very useful things such as a program to delete parts and subroutines. One program called ENVELOPE allows you to hear a simple tune and see on the screen a graph of the envelope you are creating as you change its parameters. Unfortunately, the disk is not included with the either the nine-voice or the three-voice software, but costs $49.95 extra.

HINTS & ENHANCEMENTS

From our readers

ATARI® MICROTEXT

For a quick modification to Jon Voskuil's *Microtext 1.0* (ATARI® version) to provide printing to the 820 printer, make these modifications to line 200, and add lines 2405, and 7000 through 7030.

```
200 POSITION 2,0:PRINT "SAV:ctl-S,REV:ct
1-R,LD:ctl-L,LP:ctl-P"

2405 IF C=16 THEN GOSUB 7000:REM CTL-P

7000 POKE 838,166:POKE 839,238
7010 GOSUB 3040
7020 POKE 838,163:POKE 839,246
7030 RETURN
```

This provides a format compatible command which performs quite adequately. By the way, this letter was made using this modified version. This is my first contribution to any publication, and having "ridden the bench" quite long enough, I hope this won't be my last. Thanks for the inspiration.

Bob Cowan
Pensacola, FL

ATARI® TITAN

After having received the December issue and typing in *Titan*, I was extremely impressed. Congratulations to William Morris and John Cope. After playing it several times, I could find only one area of the program that needed improvement. The idea of having to pause and type "GOTO 300" in the middle of a game does not appeal to me. So if one changes line 1280 to read:

1280 ? :? :? "GOTO 300":POSITION 2,1:POKE 764,12:ENTER "D:TITAN24K.PT2"

the computer will take care of itself. The thought behind this change is to have the computer print "GOTO 300" then place the cursor two lines above this statement, so the cursor will rest on the "G" in "GOTO" after it returns with the "READY" prompt. The "POKE 764,12" has convinced the computer that someone has pressed RETURN, so when it comes back with "READY" and the cursor is over "GOTO 300" it then prints the carriage return which, with the ATARI®'s screen editing, causes the statement to be executed; thus eliminating the user's need to type "GOTO 300". With some imagination, this idea can be quite useful. I have also used this technique with *Quest 1* and the two mazes that we now have for it.

In the January ***SoftSide***, Leonard E. Buchanan told in *Hints & Enhancements* his solution to the CLOAD sounds made by the ATARI®. The solution is quite good, but if you are like me, your TV is across the room, and I am somewhat too lazy to cross the room to turn down the volume during a CLOAD and then turn the sound back up to hear the sounds of a game. Believe it or not, Atari was thinking when they built this computer: If you POKE 65,0 you will not hear the CLOAD sounds, but you will hear game sounds. (See Atari's *BASIC Reference Manual*, Appendix I.)

Gregory A. Lovekamp
Beardstown, IL

ATARI® DATABASE

The ATARI® *Database* in the December issue is one of the best ever. It made the whole year's subscription worthwhile. I hope you will continue to publish programs of this calibre.

I would like to offer a few suggestions and raise a few questions about the *Database* program. In several places more memory can be saved. I assume that readers knew they could delete all of the REMark statements. Thanks for numbering the GOTOs and GOSUBs so this could be done.

Second, several of the REMark statements require use of the ESC key before entering. The "ctrl--" and the "shift-BACK S" are examples.

Third, some memory can be saved by deleting quotation marks at the end of program statements (e.g., lines 330-400). Using the "?" for PRINT statements where that was not done (e.g., lines 2070, 2110, 2550, and 2720) also saves a few bytes. I also notice that N1 was not used consistently to replace the number 1 (e.g., lines 1490, 1550, 1810, and 1890).

One practice I found a little annoying for such a good program was the use of a question mark and space inside quotation marks immediately before an INPUT statement (e.g., lines 2730 and 2750). These can be eliminated if the semicolon remains after the quotation marks. The failure to use the POP statement in line 1490 (and perhaps elsewhere) meant that the counter J had to be set to its maximum and a NEXT J added when POP would have accomplished the same thing. The GOTO 230 in line 210 is unnecessary, and N3 should have been used in line 2460.

Finally, note that the JJ = N0 in line 590 never is executed. It should be placed before the GOTO 610 statement.

I know these are picky matters, but I think they will improve the program. Thanks again for this very helpful program.

Michael Fink
Nashville, TN

Editor's Note: It should be pointed out that although the "?" token saves space in the line listing of a program, it does not actually save memory. All BASIC keywords are tokenized in the computer's memory, and require only one byte to store, regardless of the number of characters in the word itself.

TRS-MAN

I enjoyed the *TRS-MAN* program in the January issue of ***SoftSide***, even though I'm not ordinarily that attracted by arcade-type games. (I'm a simulations man, myself.) However, one of my kids *is* an arcade junkie, with the reflexes and eye/hand coordination of a fighter pilot. He scored "OV ERROR" on his second game!

Accordingly, I juiced up the program a little by adding bonus play and higher target scores, with an error trap for good measure; how few plays do you require to score 32,000 + ? Line 1502 helps the player keep track of how many mouths he's used up. And lines 1900-1910 are the result of a personal peeve: Games that automatically restart without asking the player's permission. The other obvious solution, incidentally, to the problem of an adolescent player who routinely racks up preposterous scores is to reduce all the point-values by a factor of ten: Make the dots worth one point, the stars ten points, and the creatures 20 to 80 points.

```
15 ONERRORGOTO1512
1502 PRINT@PO,"   ";:GOSUB1503:TR=TR+1
:PRINT@47,"MOUTHS USED: "TR;:IFTR<3THE
N1515ELSE1505
1505 IF((TR=3)AND(SC<12000))GOTO1509EL
SEIF((TR=3)ANDSC=>12000))GOTO1508ELSEI
F((TR=4)AND(SC=>18000))GOTO1508ELSEIF(
(TR=5)AND(SC=>26000))GOTO1507ELSE1509
```

```
1506 IFTR>4GOTO1509
1507 PRINT@526,"THE CHAMPION! YOUR SCO
RE WAS"SC;:PRINT@586,"AND YOU GOT THE
MAXIMUM NUMBER OF 5 PLAYS!";:FORT=1TO1
000:NEXTT:GOTO1900
1508 PRINT@525,"YOUR SCORE IS"SC" (!!
EXTRA PLAY !!)";:FORT=1TO3000:NEXTT:PR
INT@512,STRING$(63," ");:GOTO1515
1509 PRINT@534,"YOUR SCORE WAS"SC;:FOR
T=1TO1000:NEXTT:GOTO1900
1512 PRINT@526,"THE CHAMPION! YOUR SCO
RE WAS"SC;:PRINT@590,"AND YOU BROKE TH
E BANK IN"TR+1"PLAYS!";:FORT=1TO1000:N
EXTT
1513 RESUME1900
1900 PRINT@981,"ANOTHER GAME (Y/N) ";:
INPUTQ$
1910 IF Q$="N"THENCLS:ENDELSEIFQ$="Y"T
HENRUNELSE1900
```

Even though my only machine is a hopped-up TRS-80® Model III, I frequently try to work my way through the Apple and ATARI® listings, out of simple curiosity about other dialects of BASIC (though I'm not very good at translation yet). I expect you'll get another deluge of irate letters from people who prefer a single-system magazine, but I look forward to the addition of the TRS-80® Color Computer to the fold. I don't know enough about the new IBM system to judge whether there would be an audience for it or not.

Michael K. Smith
Dallas, TX

TRS-80® ONE LINER

Here is a variation on Quentin Barnes's great One Liner that appeared in your November, 1981, issue. Instead of a requiring a Restart, my version automatically clears and redraws after 100 lines are drawn.

```
1 CLS:DEFINTA-Z:RANDOM:X=32:Y=12:FORK
=1TO100:L=RND(25):XD=RND(3)-2:YD=RND(
3)-2:FORA=1TOL:SET(X,Y):SET(127-X,Y):
SET(127-X,47-Y):SET(X,47-Y):X=X+XD:Y=
Y+YD:X=X-128*INT(X/128):Y=Y-48*INT(Y/
48):NEXTA:NEXTK:GOTO1
```

Douglas A. Stuber
Columbus, OH

TRS-80® DATABASE

I look forward to the *SoftSide* each month. I think it is by far the best of the computer mags. As a retiree I have spare time to try to program, and I find lots of goodies to fool with such as *Database*, and this new *Microtext* which I am using for this masterpiece.

For what it is worth, I have been watching for correction in the multisort part of the *Database* program. It really shot down the sort for a TRS-80®. I looked at it a hundred times before finding the problem in line 8050 which read C1(0) = I-1. It should read C1(J) = I-1. If anyone is still having trouble with the sort, this should fix it.

Harry Eaton
Conneaut, OH

Editor's Note: We are planning to publish a complete, updated and enhanced *Database* program for the TRS-80® in a future issue.

DISK AND LEVEL II BASIC COMPATIBILITY

I believe that I have found a better method for making a program RUNable in both Disk and Level II BASIC. This method employs only one instruction! A simple POKE is all that's necessary to enable Disk BASIC to accept the Level II way of setting up a Machine Language routine.

First, let me explain how Disk allows for ten call addresses. When a DEFUSR statement is executed, the address following the statement is placed in a (protected) memory location corresponding to the USR number that you specified. When Disk BASIC encounters a USR call, it looks at the call number, which directly follows the USR token. It then transfers the call address from its protected memory location to locations 16526-16527, and then continues normally as Level II would.

With Disk present, locations 16809-16811 contain a JUMP instruction that directs Disk to the routine described above. When Disk is not there, they contain a RETurn instruction since the Disk routine is not present. Therefore, Level II leaves 16526-16527 alone when it makes the USR call. (That's why you have to POKE those two locations.) So if we put a RET instruction in 16809 while in Disk BASIC, the computer will act like there is no Disk present (as far as USR0-9 is concerned). And if we put RET there while in Level II, no harm is done since that's what is there already! The actual numbers are:
POKE 16809,201

Try it! While DEFUSR will not cause an error, it also will not effect where the computer branches on a USR call. You now have to POKE 16526-16527 with the address. If you're in Level II, the POKE won't do a thing. Note that this method is more permanent than the other method, since if you want to run **another** program that utilizes the DEFUSR statement, it will bomb. But note that it is not irreversible! All you have to do is restore location 16809 with 195, and Disk will act as usual. (Doing this in Level II will cause a major program bomb when you try to make a USR call!) Also note that CMD"T" is not effected, and will still have to be executed where sound is used.

Despite its problems, this method saves a lot of code. While it is just as easy to use the old method in sound routines, any other application will work correctly under my method. The POKE should probably be put in the beginning along with the POKE that fixes the data error (POKE 16553,255). I hope this information is useful to anyone interested in making their programs Disk/Level II compatible.

David Yon
Richmond, VT

When does your *SoftSide* subscription expire?

```
49007STANTG97*B00812@1OCT 82
1102691 017                M
GEOFFREY STANTON
97 BAYSIDE COURT
KALAMAZOO, MICH. 49007
```

The last five characters (three letters for month, two numbers for year) on the top line of your mailing label will tell you when your subscription ends.

For more information, write:
SoftSide
515 Abbot Drive, Broomall PA 19008

Bugs, Worms

and other undesirables

The TRS-80® version of *Microtext 1.1* (January, 1982) suffers from an omission in the line listing. The coding necessary for the Model I to insert an "@" symbol in front of a letter which is to be capitalized was left out. Line 530 needs to be added, and line 740 modified just slightly as shown below.

```
530 IF C>90 THEN PRINT B$;"@ ";: L$(LN
)=L$(LN)+"@": CHAR=CHAR+1: C$=CHR$(C-3
2): GOTO 740

740 CHAR=CHAR+1: IF CHAR>=LWID AND C<>
SPC AND C<>RTN THEN GOSUB 1000
```

Notice that this change is necessary only for the Model I, and not for the Model III.

In the ATARI® *Database* program (December, 1981), the lines 1700 and 1705 on page 71 should replace line 1700 which was listed in the program.

CALENDAR

March 1-4

Robots VI Conference and Exposition
Cobo Hall, Detroit, MI

Featuring the latest in robot technology and equipment, this conference will deal with topics such as assembly, aerospace applications, vision and handling, research and development, and various other subjects. The show is sponsored by Robotics International of the Society of Manufacturing Engineers (RI/SME).
Contact: RI/SME, One SME Dr., P.O. Box 930, Dearborn, MI 48128, (313) 271-1500, ext. 416

March 1-3

National Conference on Publishing and Printing: Technology and Management
Center for Technology and Administration, American University, Washington, DC

This conference will cover the latest technological developments, trends and management decision-making methods for the publishing, printing and information industries.
Contact: Prof. Lowell Hattery, Center for Technology and Administration, American University, Washington, DC, (202) 686-2513; Registration: U.S. Professional Development Institute, 12611 Davan Dr., Silver Spring, MD 20904, (301) 622-0066

March 1-3; 4-6

Virginia Tech Chemathon Workshops
Virginia Polytechnic Institute and State University, Blacksburg, VA

1. Microcomputer Interfacing Design and Programming: $395.
2. Personal Computers for Instrument Automation: $395.
Contact: Dr. Linda Leffel, C.E.C., VPI and SU, Blacksburg, VA 24061, (703) 961-4848.

March 9-11

The 1982 International Zurich Seminar on Digital Communications
Zurich, Switzerland

The theme is "Man-Machine Interaction."
Contact: Secretariat '82 IZS, M. Frey, EAE, Siemens-Albis AG, POB CH-8047, Zurich, Switzerland

March 12-13

Computers in Education Conference
Seattle Pacific University, Seattle, WA

Talks, workshops, and exhibits with emphasis on the use of microcomputer in K-12 classrooms of various disciplines.
Contact: Tony Jongejan, Everett High School, 2416 Colby, Everett, WA 98201.

March 18-20; 24-26

Microcomputers in Education
Mesa Community College, Phoenix, AZ; Lewis and Clark College, Portland, OR

Designed for professional development for educators at all levels. Topics include: Overview, Logo, BASIC, Graphics, Pascal, Micros in Math, Science, and Administration. Offered by Technical Education Research Centers, Inc. (TERC).
Contact: TERC, 8 Eliot St., Cambridge, MA 02138

March 19-21

7th Annual West Coast Computer Faire
Civic Auditorium and Brooks Hall, San Francisco, CA

Over 300 exhibitors and a wide assortment of seminars. The grandaddy of computer shows.
Contact: The Computer Faire, 333 Swett Rd., Woodside, CA 94062, (415) 851-7075

March 20-21

1982 Charlotte Hamfest and Computerfair
Charlotte Civic Center, Charlotte, NC

For amateur radio and computer hobbyists.
Contact: Lane Tarleton, Charlotte Hamfest and Computerfair, Mecklenburg Amateur Radio Society, 2425 Park Rd., Charlotte, NC 28203, (800) 845-6183

March 22-25

Interface '82
Dallas Convention Center, Dallas, TX

Co-sponsored by The Interface Group, and MacGraw-Hill's *Business Week* and *Data Communications* magazines, this data communications/information processing conference and exposition is for end-users.

March 28-31

Electronic Banking Conference and Equipment Exposition
Hyatt Hotel, Los Angeles CA

Features the latest developments in electronic banking and equipment, including hardware, software, systems, and services.
Contact: Karen Hayes, The Bankers Institute, 21 Tamal Blvd., Corte Madera, CA 94925, (415) 924-1420

April 2-4

The Eighty/Apple Computer Show.
New York Statler Hotel, New York, NY

Devoted to the TRS-80® and Apple computers, this show features hardware, software, books, magazines, and accessories, as well as over 100 exhibitors.
Contact: Ken Gordon, Kengore Corporation, 3001 Route 27, Franklin Park, NJ 08823, (201) 297-2526

If you or your organization are sponsoring or know of an event you think would be of interest to *SoftSide* readers, please send complete information to:

SoftSide Publications
Calendar Editor
6 South Street
Milford, NH 03055

Be sure to include complete information concerning dates, location, subject matter and a contact name, address, and phone number. Please submit material two months prior to the date of the event. Thank you.

SoftSide Selections Ordering Information

USA Orders

SoftSide Selections accepts VISA, MasterCard, Certified Checks, Money Orders and Personal Checks. **SoftSide** Selections pays all shipping charges on domestic PREPAID orders OVER $100. On all PREPAID orders under $100 a handling charge of $2.50 must be added.

C.O.D.

C.O.D. orders accepted for U.S. shipment only. There is a $2.50 ADDITIONAL C.O.D. charge.

Canada/Mexico

No C.O.D. to Canada or Mexico. The preferred method of payment is by MasterCard or VISA. NO PERSONAL OR COMPANY CHECKS. A bank check is acceptable if it has been preprinted for payment in U.S. dollars. The handling charge on all Canadian or Mexican orders is $5.00 PLUS actual shipping charges.

Other Foreign Orders

Payment must either be by a BANK CHECK drawn on a U.S. bank, payable in U.S. dollars or by affiliated bank credit cards of VISA or MasterCard. All shipping and duty charges are the customer's responsibility. All overseas orders are subject to a $10.00 handling charge PLUS actual postage charges.

Guarantee

All software is guaranteed to load and run. If you experience difficulties with the product within 30 days, the tape or disk may be returned. Call (603)673-0586 for a Return Authorization Number. Any returns without a Return Authorization Number clearly marked on the outside WILL BE REFUSED. Send your properly protected disk or tape to the attention of Customer Service Representative with a note including your name and address.

Liability

All software is sold on an as-is basis. **SoftSide** assumes *no liability* for loss or damage caused or alleged to be caused directly or indirectly by products sold or exchanged by them or their distributors, including, but not limited to, any interruption in service, loss of business or anticipatory profits or consequential damages resulting from use or operation of such software.

Prices

Prices are subject to change without notice. We are not responsible for typographical errors.

OUTGOING MAIL

by Randal L. Kottwitz

I've been looking through the collection of English dictionaries in the reference library here at ***SoftSide*** and cannot find a single definition for the word "database" in any of its various forms. True, there are definitions given in almost any computer book with a glossary, but those definitions vary so greatly that there seems to be no clear understanding, even within the industry which uses the term, as to what a "database" is.

As ***SoftSide*** prepares to publish various databases on DV and, in the future, on-line, I feel it is necessary we come to an understanding of just exactly what it is we're publishing. In addition, as the Envyrn™ project progresses, its approach to the organization, storage and retrieval of data demands a thorough understanding of our current methods of handling data.

If you consider a database to be a group of data, compiled in such a manner as to allow ease of future access, you might consider such collections as book libraries, record and audio tape collections, video disc and tape collections, and even stamp and coin collections as forms of databases. A more restrictive view might be that in order to be classified as a database, information must be able to reside in the memory of a computer. Obviously, the latter view is the one more applicable to ***SoftSide*** as we now know it. However, it is vital that, with an eye on the future, we explore all of the possibilities for data storage and access.

As the interface of computer and video disc comes closer to a realistic option every day, yet another view of data storage and retreival is coming into view for computerists. Other fields are facing similar problems in dealing with some of the same hardware and we'd be wise to keep a watchful eye on their solutions' relativity to our applications.

One of the rumored video disc applications now in the development stages is being carried on by one of the major art museums in this country. They are attempting to store an image of every canvas and sculpture in their vast collection on *one* video disc! As nearly as I can estimate, the project involves somewhere between thirty and forty thousand images. The organization problems inherent in the project are staggering. It will be wonderful to have such a compact library of images available at the touch of a button, but how will you choose which of the forty thousand available "frames" you wish to view? Certainly a printed volume acting as a catalog is an absolute must, but by what criteria should such a catalog be organized? I may prefer to choose my program of images by the artist, but the fellow down the street might very well wish to choose his according to the primary color of the images. I have no proposed solution to the difficulty this presents, I only wish to point to the very problem of organization our society is facing. As we develop the technology to make vast amounts of data readily accessible, we then face an entirely different problem — how do we choose which data we wish to read, view, listen or experience?

It is at the point of choice that the database manager enters the picture. The job of the database manager is to allow the chooser to make a choice of data and criteria by which to experience it, with the least amount of aggravation. I shudder to imagine having to look through an alphabetic listing of all of the paintings and sculptures by *title* on the video disc I mentioned, when my goal is to view a favorite painting by Degas whose title I can't recall. An efficient database manager should allow me to choose the criteria by which I search for a piece of data as well as the specific data I wish to experience. The database managers currently on the market for the microcomputer certainly give us a more complex manner in which to handle data than we've ever experienced before, but I fear that their methods of data manipulation are not progressing as rapidly as the accumulation of data for them to manipulate.

As life becomes more complex at an ever increasing speed, the human being is demanding access to the knowledge believed to be necessary to stay on "the leading edge." Computerists are closer to that "edge" than the vast majority of the public. However, I must wonder if we are gaining a thorough enough understanding of the unique points of view the keyboard in front of us allows. Someday, we may very well be issued a datapak the size of a man's wallet which would contain all the information in the entire Library of Congress. We then will be forced to contemplate the applicability of the Dewey Decimal System, one designed for paper cards in drawers, as opposed to a system designed for use from a terminal which accesses the information as well as the key to finding it. The time for such contemplation is now, not after the conversion of our great databases from paper to electronic media has already taken place. We must maintain our perspective of the body of human knowledge as well as the raw data of which it is composed.

Until next month, happy hacking!

Randal L. Kottwitz

Randal L. Kottwitz
Associate Publisher/Editor

Your Adventures

June Adventure of the Month
Arabian Adventure

As Sinbad, the mightiest sailor in ancient Arabia, your mission is to rescue Princess Jasmine from the clutches of the Wizard of Darkness. You will cross the Seven Seas to the deadly Cyclops Mountain, and do battle with skeletons, a one-eyed beast, a hairy tarantula and more monsters who try to thwart your noble pursuit.

July Adventure of the Month
Alien Adventure

You are the sole survivor of a crew on a mission to deliver a cargo of oil to Earth. A crash landing has left you stranded on a small planet, harshly alien but rich in lead, gold and platinum. You must find provisions and a means of leaving the planet. But beware of the THING that massacred your crew!

August Adventure of the Month
Treasure Island Adventure

You are a hardy adventurer in search of fame, fortune, and whatever else you can get. You find yourself on an island where there is rumor of pirate's treasure. But watch out for the evil magician and the underground torture chamber! You may end up in a spot where all roads coming into it are paved with good intentions. . .

September Adventure of the Month
Jack The Ripper Adventure

Jack the Ripper is running rampant in London and you must stop him! Scotland Yard demands that you take action, and the only answer is to set yourself up as a decoy. Be careful how you plan your costume, or dear Jack will laugh hysterically and leave you in the dust!

October Adventure of the Month
Crime Adventure

Test your skills as a detective by sifting through hundreds of clues. You may have to become the new Sherlock Holmes to solve this one! Look for the strange, but don't overlook the obvious, as you try to find Mrs. Fenwick and return her to where she belongs.

November Adventure of the Month
Around the World in Eighty Days Adventure

Try to repeat the feat of the classic novel, complete with a balloon and other exciting features of the original adventure. Are you ready to take the challenge? Bon voyage!

December Adventure of the Month
Black Hole Adventure

The crew of an interstellar craft discovers the long-lost Deep-Space Probe One, the Cygnus, at the edge of the vortex surrounding an immense black hole. See if you can foil the plans of Dr. Hans Reinhardt.

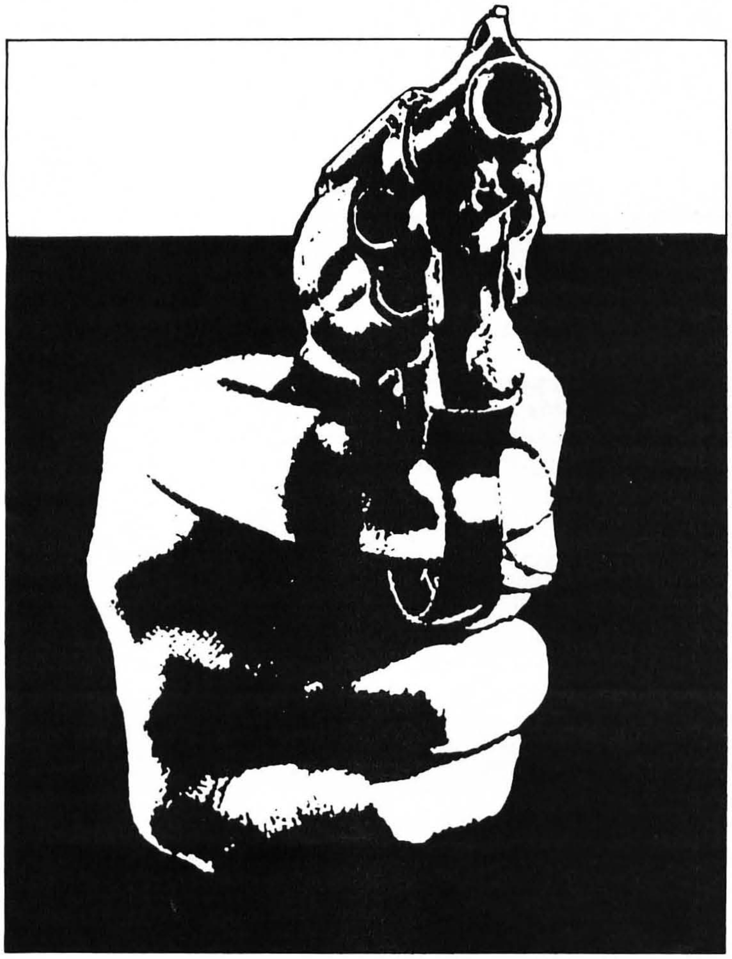
January Adventure of the Month
Windsloe Mansion Adventure

A famous prisoner lies in the dungeon of an old mansion. An underground passage connects with the Blair house, whose owners will help you to rescue the prisoner. Can you overcome the human and supernatural creatures who inhabit Windsloe Mansion?

February Adventure of the Month
Klondike Adventure

Snow, ice, and bitter cold surround you. Your search for fame and fortune in the northern country will lead you through many perils, but you may also see some familiar faces along the way. This breezy adventure will keep you occupied inside while the winter winds blow outdoors.

will start here —



MARCH ADVENTURE OF THE MONTH
JAMES BRAND ADVENTURE

The President's life is in danger. As James Brand, you must save his life and destroy the evil Dr. Death. Your life is constantly on the line; each move you make could be your last. "Your assignment, Mr. Brand...."

Subscribe to Adventure of the Month

How would you like to go back in time to 19th century London to match wits with Jack the Ripper? Out into space to brave the swirling vortex of a black hole? Into the depths of the ocean, or on a quest to rescue a beautiful princess from the clutches of evil monsters?

You never know where **SoftSide's Adventure of the Month** might take you. But you can be sure that each month you will experience new delights and new challenges as you receive an original adventure on tape or disk, ready to load into your computer.

The cost? A six-month membership is just $27 for the tape ($4.50 per adventure) or $45 for the disk ($7.50 per adventure). If you're not sure that you can take six full months of excitement, you can order a single tape for $6 or a disk for $9. Or, if you're especially adventuresome, we're offering disks, each packed with three great adventures, for only $24 per disk.

Please use the coupon below (or the bind-in card in this issue) to order.

Adventure of the Month
6 South Street, Milford NH 03055

Yes, I'm ready to start! Send me Adventures —

- **Six month subscription:**
  - ☐ Cassette ($27)
  - ☐ Disk ($45)
- **Individual adventures (please specify)**

  ____
  ____
  ____
  - ☐ Cassette — $6 each
  - ☐ Disk — $9 each
- **Three adventures on one super disk ($24 each):**
  - ☐ Arabian, Alien, & Treasure Island Adventures
  - ☐ Jack the Ripper, Crime, & Around the World Adventures
  - ☐ Black Hole, Windsloe Mansion, & Klondike Adventures

Please specify which computer:
- ☐ Apple (req. 24K for tape, 32K for disk)
- ☐ ATARI® (req. 32K for tape, 40K for disk)
- ☐ TRS-80® (req. 16K for tape, 32K for disk)

Name ______________________
Address ______________________
City/State ______________________ Zip __________
☐ Payment enclosed

☐ MasterCard ☐ VISA Name of Cardholder ______________
MC# and Interbank#/VISA# ______________
Exp. Date ________ Signature ______________

Prices subject to change without notice. Apple, ATARI* and TRS-80* are registered trademarks of The Apple Computer Company, Warner Communications and The Tandy Corporation respectively.

The Third Dimension

By Allen L. Wold and Fred D'Ignazio

C3PO and Chewbacca sit across a round table, the top of which is divided into concentric segments. Holographic chess pieces, in the form of strange alien creatures, move at the push of a button. They march across the "squares," threaten each other, and fight animatedly.

But it's all done with trick photography.

Still, we recognize the pieces as holographic images, projected somehow onto the surface of the table. Holographic techniques are improving rapidly, and someday a game very much like that will be possible — won't it?

Unfortunately, it's not a matter of technology, but of physics. The game in *Star Wars* is not likely ever to be realized. To find out why not, let us take a very brief look at how holograms are made, look at some alternatives, and then see what holography can do.

Holograms are produced by laser light, which is both extremely monochromatic (single-color), and focused so that the rays are all parallel to each other. A beam of laser light is split in two. One beam, the reference beam, shines directly on the holographic plate (very much like regular photographic film). The other beam shines on the object to be reproduced, and is reflected onto the plate.

When the direct and the reflected beams strike the plate, they interfere with each other, producing a very complex and visually unintelligible pattern. However, when the developed plate is later illuminated by a laser beam similar to the one by which the plate was exposed, a three-dimensional image is produced.

The technology to make holograms is quite complicated, and further discussion is out of place here. Suffice it to say that there are basically two kinds of images — those which appear behind the plate, and those which seem to stand out in front of the plate.

If you're interested in the subject, one good book is *Understanding Holography* by Michael Wenyon (NY Arco, 1978).

Let us first look at projection holography, where the image is made to appear on a table top or other surface, as was represented in *Star Wars*. This is quite possible today, but is subject to certain physical limitations inherent in the nature of holography, rather than in the technology used to produce it.

The first problem is that although images can be projected onto a flat surface (or into mid-air for that matter), when they are viewed from any angle other than that by which the light leaves the plate, they aren't there. While certain forms of transmission holography, involving a cylindrical plate, allow you to see an image from many directions, you can't see the image of a projected hologram if you are even slightly out of line.

Secondly, the size of a holographic image is dependent on the size of the plate which produces it. The image can be no larger than the plate itself, though it can be viewed through magnifiers. Still, with magnifiers between the viewer and the image, it is as if the image were in a TV box, not standing above a table. And the magnifiers must be nearly as large as the desired image. This is true whether the image is projected, or seen through the plate.

The plate size also determines how far the image can be projected. A plate 12 inches on a side will project the image about six inches in front of it. To project an image six feet away would require a plate nearly 12 feet across.

Thus, the size, distance, and viewing angle of the images are directly related to the size of the film, and the angle through which the light passes to form the image.

This leaves us with looking at holographic images in a TV-like box, instead of floating free. Here technology can do something about the size of the image formed. There is no reason why a hologram couldn't be as big as a living room wall. And while most holograms today are monochromatic, they can be made in color by projecting three images as is done in color television. But there are still problems and limits. For example, while it might be relatively easy to make a holographic movie, holographic TV is something else.

TV images are broadcast at the rate of 30 per second. Each image contains considerable information, thousands of dots of light, in thousands of differing levels of brightness. A hologram, by its physical structure, contains about 300,000 times as much information as a TV picture of the same size. The time required to produce a one inch by one inch image is, at present, about two hours. A hologram large enough to fill the screen of a 19-inch TV would require several times as long to produce. And that would be for just a single "frame," not the 30 frames per second required for the same quality of illusion of movement that a TV would produce.

For the moment, we'll have to content ourselves with prerecorded holograms instead of live broadcasts. This does not take into consideration the difficulty of making a hologram in the first place, which requires absolute darkness except for the laser light illuminating the subject. This is not to say that such a broadcast will be impossible in the future, but it is the far future indeed.

Before we go on with holograms, let us look at some alternatives. One possibility is a system of computer controlled 3-D images which makes use of a rapidly moving mirror, which reflects a different layer of space-filling image at each position. The mirror moves so quickly that there is no flicker, but the systems cost around $100,000.

Some electronic games available today use perspective drawings on an LCD display, such as *Escape 1000 Mazes*, or the arcade game *Red Baron*, which uses similar drawings on a CRT. Many computer games make use of perspective drawings to enhance the sense of three-dimensionality. If there is color available, color code could be used to enhance the sense of depth — the redder, the closer; the bluer, the farther away. But this is only a representation, not a true 3-D image.

And there is the old standby, the stereoscope, where the viewer uses special glasses so that two slightly different images are seen each with only one eye. The same principle was used with the so-called 3-D movies. But again it is only an illusion. One's perspective does not change as one views the image from different angles.

Chess can be played in three dimensions. One way is to use one board for each level, laid side by side. But it is hard to perceive the 3-D relationships. (You can even play four-dimensional chess this way, which is even more difficult.) A game on the market uses three boards, each 8 x 8, suspended one above the other by a rack at one side. One author, Allen, has a "board," which consists of five surfaces, each divided into 5 x 5 squares, forming a cube. An 8 x 8 board would be equally possible.

There are a number of board games on the market representing conflict in space. In some of these the strategy is emphasized, as in *Imperium* or *Triplanetary*, and three-dimensionality is relatively unimportant. Several others, however, stress the tactics of combat. *Battlefleet Mars*, for example, has a double board on which three dimensions are represented. *Starforce*, on the other hand, represents the whole cluster of nearby stars on a flat field, with special rules for simulating the third dimension.

Space is three-dimensional, and flat playing surfaces represent this only imperfectly. The problems of keeping track, by using two boards for XY and XZ planes, or code letters representing levels above or below the playing surface, are not very visual, and interfere with play.

While three-dimensional boards, such as for chess, might be built, space games usually require more than five or eight spaces in any one direction. Thirty or 40 units in each dimension is more typical. But such a board, constructed of card or transparent plastic, would be too cumbersome to use. How do we move that piece right in the middle, with tongs?

At the moment, the only solution is to produce the images in a kind of tank, or TV-like box, permitting viewing only from the front. The inside is hollow, and at the sides are the various holographic plates and the source of laser light which produces the images. Thus, you see the images as if you were looking through a window.

Whatever game is to be played using this device, we assume that all tables, moves, combat, and so forth are controlled by, or at least moderated by a computer, as we described in an earlier column. There are no dice to roll, no combat tables to look up. The players don't actually touch the pieces. They simply key in the desired movement or action, and the computer performs it.

One other holographic trick must be mentioned, and that is that a single holographic plate can contain up to 100 separate images. When the plate is illuminated, only one image is seen, depending on the angle of the light. To see another image, the light is moved slightly, and a different image appears.

Under computer control, a set of holographic images could be presented according to the outcome of game decisions.

Thus, under computer control, a predetermined set of images could be presented in any order, according to the outcome of game decisions. Several different plates, each with a hundred images, could be switched back and forth in front of the laser.

In a space game, one holographic plate could contain the images of stars in their relative positions. Seen through the TV-like window, they would appear truly three-dimensional. As the player's position in space changed, different star images would be produced, giving the view as it might be from any new location.

Another plate could contain images of starships, which would be superimposed over the star-field image. As the ships move relative to the stars, the computer would present different images representing their different positions. This might require one plate per ship, but since the image would be small, a frame could hold dozens of holograms, each one with a hundred ship positions.

Holographic plates can be made of plastic, and mass-produced, and can be made quite large. The expense of production would be subject to technological development and marketing, and could conceivably be brought to reasonable levels. One might then see a game like *Star Force*, in which one sat before a rather large box, and saw all the stars within 15 light years displayed in natural color and proper brightness to indicate size, within the black volume of space.

Fleets or single ships would be tiny spots of light, positioned among the stars, and moving, according to the player's instructions, in a truly three-dimensional fashion. Whenever ship-to-ship combat was desired, the computer would store the position of all ships and present an image at a much smaller scale, representing perhaps only one light minute on a side. Any type of ship movement would be possible, and aiming and firing, maneuvering and damage, would all be produced using a second set of holographic plates.

A solitaire game of space combat could be produced in which the player's view is from the bridge or cockpit of his fighter, similar to that of *Red Baron*, except the image would be truly three-dimensional. What the player would see is what's "out front." Computer-controlled enemies would move, and the player would have a true 3-D image of enemy vessels flashing past, his missiles and beams searing off through space.

But we don't have to limit ourselves to space games. *Dungeons & Dragons* type adventure games could easily take advantage of 3-D holographic imaging. The restriction to a box is no problem, since most of the action in such games takes place in a room or section of dungeon anyway. Outdoor adventures would be more difficult to produce, but not impossible.

As in the space games, one set of holographic plates could produce the images of the environment, a second set would display the adventurers, with perhaps a third set for the monsters, treasures, and movable furniture.

Full motion would require an excessive number of image plates, but "stop-action" scenes would be easy. Your hero could be in any part of the room, in any position, and the computer would select which image to present depending on your instructions. It could display a magician casting a spell, warriors fighting a troll, and accommodate the player-characters carting off treasure.

Holographic imaging is not — yet — the perfect answer to truly three-dimensional games, but even with its limits, there is a lot of potential.

Envyrn™

Putting micro-graphics into action

Micrographics are now practical. With Envyrn™, maps, charts, graphs, floor plans, or any physical situation may be created or duplicated — in three levels of resolution. After you've created your graphics, save them, redesign them, create hard copy, and put them into action. With the artificial intelligence features of Envyrn™, you can make your graphics "smart," with prescribed reactions and information hidden under the display. You can truly create your own world.

In the Envyrn™ Participation Program, you will be a part of the documented experience in this new software innovation. During your full year of participation, you will receive the initial Envyrn™ editor (a much expanded version of the one published in SoftSide), extensive supporting documentation, program notes, suggested applications and at least one completely updated editor later in the year — all for only $200. Envyrn™ is destined to become one of the standard utilities in your program library. Take part in this unique experience. The Envyrn™ Participation Program — Putting micrographics into action.

Envyrn™

Participation Program

6 South Street, Milford, NH 03055

☐ **One-year subscription to the Envyrn™ Participation Program — $200.00**

Name______________________________

Address______________________________

City/State __________________ Zip__________

☐ Payment enclosed

☐ MasterCard ☐ VISA Exp.Date__________

Name of Cardholder ______________________________

MC# and Interbank#/VISA#______________________________

Signature ______________________________

I own a 48K TRS-80® with Disk ☐ Model I ☐ Model III

TRS-80 is a registered trademark of Tandy Corporation

diversions™

thru Envyrn

Module One

PARSEC™

You hover on the fringe, five parsecs from initialization of your mission. Your culture is tired and in need of new vitality. The Chamber can duplicate the essence of life from any species and you're about to start your "collection." What fate awaits you on this unique mission? Is it true that cultures opposed to your sampling have been preparing their defenses and stand ready to resist your intrusions?

The first module is prepared for take-off — **diversions thru Envyrn**™ presents **Parsec**™, a graphics space adventure unlike any you've ever played. **Parsec**™, created with the **Envyrn**™ graphics editor, simulates deep space in your TRS-80® computer, creating an action-filled field of play larger than any other microcomputer game. **Parsec**™ takes you beyond the screen and into worlds not yet seen by you or your computer. Join the adventure — subscribe to **diversions thru Envyrn**™ now or send $20 for **Parsec**™, the first of six modules to be released in 1982. Each module includes a magazine and a disk.

diversions™

thru Envyrn

6 South Street Milford, NH 03055

☐ One-year subscription (6 modules) — $60.00

☐ Sample **Envyrnment**™ Parsec™ only — $20.00

Name ____________________

Address ____________________

City/State ____________________ Zip ________

☐ Payment enclosed

☐ MasterCard ☐ VISA Exp.Date ________

Name of Cardholder ____________________

MC# and Interbank#/VISA# ____________________

Signature ____________________

To order use the card provided in this issue or send in this order blank.

I own a 48K TRS-80® with Disk ☐ Model I ☐ Model III

TRS-80 is a registered trademark of Tandy Corporation

The Eleventh Hour

By "J"

"To boldly go where no man has gone before . . . !"

Anyone who doesn't know the origin of this quotation has no business reading a computer magazine. And especially not *The Sensuous Programmer*. And especially not this particular installment, wherein I venture into a world where I've been telling myself for ten months that I'd never go.

Nevertheless, the letter has flooded in requesting that I write about this subject, so I have decided to forsake all my well-reasoned vows of silence and rush (creep?) in where angels fear to tread.

The topic is graphics. How could I have deceived myself by thinking that I could write a 12-column series (yes, folks, next month is the last) without a discussion of graphics? Yet this is the one area where the three computers we're discussing are most different from one another. And because of that, this article will be the most basic of introductions, skipping over anything that would lure me out to drift in the vast wine-dark sea of my ignorance.

TRS-80® Graphics

Since TRS-80® graphics are the least complex of the three, let's start with this venerable machine. Part of the simplicity here, of course, lies in the fact that there are only two "colors" available: black and white. A given picture element (pixel) is either on or off; there are no further parameters to be specified. There are two different methods, however, for specifying which pixels shall be on or off: using the SET and RESET statements, and using character graphics.

The TRS-80®'s screen is divided into 48 rows of 128 pixels each. That makes a total of 6144 pixels, each of which is about twice as high as it is wide. Each of these pixels can be selectively turned on or off using the BASIC statements SET and RESET. The sample program below clears the screen (RESETs all the pixels) using CLS; then SETs a horizontal line of pixels from one edge of the screen to the other, midway down; and then RESETs three of those, breaking the line into four segments.

```
100 CLS
110 FOR X=0 TO 127
120 SET (X,23)
130 NEXT X
140 RESET (10,23)
150 RESET (30,23)
160 RESET (70,23)
```

Notice that horizontal lines drawn one pixel high will be considerably thicker than vertical lines drawn one pixel wide, because of the shape of the pixel. Two pixels SET side-by-side form a nearly square shape; thus a vertical line two pixels wide will be about the same thickness as a horizontal line one pixel high.

Any graphics shape that can be represented using a 128 by 48 grid of bricks-standing-on-end can thus be drawn on the TRS-80® screen using SET and RESET. Building a picture in this way can, however, be tedious and time-consuming if it's fairly complex.

There is a second way of doing the same thing, which often has significant advantages over the SET/RESET method. This second method uses a slightly different way of visualizing the screen. Instead of seeing the screen as 6144 individual pixels, it sees it as 1024 blocks, each containing six pixels. There are, then, 16 rows of 64 blocks, each of them two pixels wide and three high.

Now, the idea is that the six pixels in each of these blocks can be arranged in 64 different on/off patterns. Pattern #1 has only the upper left pixel on; pattern #2 has only the upper right pixel on; pattern #57 has the upper left, middle right, and lower left and right pixels on; and so forth. Thus patterns 0 (all off) through 63 (all on) can represent all the unique combinations of the six pixels.

It's no accident that each of these graphics blocks occupies the same screen space as a single text character (plus the between-lines blank space which the text character doesn't actually use). This makes it possible to PRINT a graphics block just as a text character is PRINTed, at any of the 1024 character positions on the screen. All you have to do is to add 128 to the "pattern number" of the graphics block, and PRINT the character having that ASCII value wherever you want it.

For example, the following program line will PRINT a solid white block (all six pixels on) at the first character position on the tenth line of the screen:

```
200 PRINT@ 640, CHR$(191)
```

This would have the same effect as SETting the six pixels located at positions (0,31), (1,31), (0,32), (1,32), (0,33), and (1,33).

The advantages of this method become apparent when you realize that you can assign a string of graphics characters (along with such goodies as linefeeds, backspaces, carriage returns, and normal text characters) to a string variable, and then simply PRINT that string at any screen position. A whole screenful of graphics can thus be stored in no more than four or five strings, and PRINTed much (much!) faster than would be possible with the SET/RESET method.

Apple Graphics

The Apple II has two different graphics modes. One has fewer pixels than does the TRS-80®; the other has a great deal more. In both cases, the approach to displaying graphics on the screen is rather different than with the TRS-80®.

In the low-resolution graphics mode, the Apple screen is divided into 48 rows of 40 pixels each — the same number of rows as the TRS-80®, but with each pixel in the row about three times as wide. (Overall, the width is about one and a half times the height.) Graphics pixels and text characters cannot be intermixed on the same part of the screen at the same time, but the screen can be (and normally is) split into an upper portion of 40 rows of graphics pixels and a lower portion of four rows of text. Each of the pixels can be set to any one of 16 different colors, including black and white.

There are five BASIC statements used to create graphics in the low-resolution mode. The first one is used to change the normal text display to the graphics display mode. Issuing the command GR, or using it in a program line, leaves the four bottom text lines of the screen intact, sets the remainder of the screen to low-resolution graphics mode, and clears the 1600 graphics pixels to black. (In this mode the HOME command, which normally clears the whole screen of text, will clear only the four text lines.)

The PLOT statement then allows you to change the color of individual pixels. It functions very much like the SET and RESET statements on the TRS-80®, except that the coordinates are not enclosed in parentheses, and the color of the pixel must be specified prior to the PLOT using the COLOR= statement. Legitimate COLOR numbers range from 0 through 15: 0 is black and 15 is white, with a variety of colors in between. (Actually, numbers can range up to 255, but the colors simply repeat themselves.)

In addition to PLOTting individual pixels, the Apple allows you an easy method of drawing solid rows or columns of blocks. The statements VLIN...AT and HLIN...AT plot vertical and horizontal lines on the low-resolution screen. These statements require three coordinates to be given: the starting and ending row or column for the line, and the column or row in which it is to be drawn.

The following example should help to clarify Apple low-resolution graphics. It will have an effect similar to the prior example for the TRS-80®. It first clears the upper part of the screen using GR; then plots a Horizontal LINe of pixels from one edge of the screen to the other, midway down, using COLOR #2 (dark blue); and then PLOTs three individual yellow pixels in the line (COLOR #13).

```
300 GR
310 COLOR= 2
320 HLIN 0,39 AT 20
330 COLOR= 13
340 PLOT 5,20
350 PLOT 10,20
360 PLOT 25,20
```

The other Apple graphics mode gives greatly increased resolution. Instead of a mere 1920 bricks (full screen), high-resolution graphics gives you 53,760 dots in the same area: 28 high-resolution dots for each low-resolution block. These are arranged in 192 rows of 280 pixels each. Again, graphics and text cannot be intermixed, but the bottom part of the screen can be reserved for four lines of text — limiting the graphics display to 160 lines. The available colors are limited to eight, of which, only six are actually unique: green, violet, orange, blue, two whites, and two blacks. Because of the way the pixels are stored in memory, the first four colors can actually be plotted only in alternate columns: green and orange in odd columns, and violet and blue in even columns.

Four BASIC statements govern the use of high-resolution graphics. Two different memory areas can be used for high-resolution screen memory; the first one is accessed by the statement HGR, and the second by HGR2. HGR defaults to a mixed graphics/text mode, as does the low-resolution GR; both can be changed to full-screen using a memory POKE. HGR2 defaults to full-screen graphics; it can be POKEd into a mixed-screen mode, but the four text lines at the bottom are not from normal text memory and are clumsy to access.

A pixel or series of pixels is plotted in a given color using the HCOLOR= and HPLOT statements. The HCOLOR number must be in the range of 0 through 7 (where 0 and 4 are black, and 3 and 7 are white). HPLOT can be used to plot individual pixels, or to draw straight lines between any specified points. The various formats for the statement are as follows:

```
HPLOT x1,y1
HPLOT TO x2,y2 (TO x3,y3 TO ...)
HPLOT x4,y4 TO x5,y5 (TO ...)
```

The parentheses are NOT part of the syntax; they simply indicate optional continuations of the statements. The middle statement draws the line from the last HPLOTted point, continuing with the same color — even if a different color has been specified in the meantime using the HCOLOR= statement.

An additional method exists for manipulating high-resolution graphics, which involves creating an encoded "shape table" in memory and then drawing the shape(s) on the screen. The method for creating the shape table is rather involved but, once done, enables very fast manipulation of the stored shapes. They can be DRAWn or XDRAWn (each point reversed to its own complementary color) at any screen location, enlarged and shrunk using the SCALE= statement, and rotated using the ROT= statement. (Editor's note: See Thomas Keith's *Magical Shape Machine* utility elsewhere in this issue.)

ATARI® Graphics

Graphics is an area in which the ATARI® really shines. Its graphics capabilities are more varied than either the TRS-80® or the Apple, and this introduction will leave even more unsaid than will the others. Six different graphics modes, plus two specialized text modes, are available in addition to the normal text mode. These offer various degrees of resolution and color versatility, with corresponding memory requirements.

There are 16 different colors available, each of them in eight different levels of luminance (lightness), for a total of 128 distinct combinations. However, only a few of these can be displayed on the screen at the same time using normal graphics commands. Depending on the graphics mode chosen, up to four different color/luminance combinations can be used at once.

Each of the desired combinations is defined using the SETCOLOR statement, which assigns a certain color and luminance to one of five color registers. Its format is SETCOLOR r,c,l where r is the register number, c is the color number, and l is the luminance number. You then specify which color register you want the computer to use in drawing on the screen, using the COLOR statement (e.g., *COLOR 2*). To confuse things somewhat, the number used in the COLOR statement is usually not the same as the color register to which it refers; a chart on page 53 of the *ATARI® BASIC Reference Manual* gives the details when needed.

Resolution varies from 24 rows of 40 pixels each (960 total), to 192 rows of 320 pixels each (61,440 total). These figures are for full-screen graphics. Like the Apple, the ATARI® allows you to reserve four normal text lines at the bottom of the screen. The chart in Figure 1 shows the resolution and number of colors available in each of the six graphics modes. (Modes 0, 1, and 2 are text modes: 0 is the "normal" text mode, and 1 and 2 allow printing enlarged text characters on the screen.)

Although the number of pixels given is for full-screen graphics, each of the modes shown actually defaults to the split-screen format. Full-screen display is achieved by adding 16 to the value of the graphics mode. Adding 16 more to either split-screen or full-screen mode numbers will prevent the screen from being cleared when the GRAPHICS statement is executed.

The desired graphics mode is chosen using the GRAPHICS statement. Its format is simply GRAPHICS n, where n is the number of the mode. (It's possible, by the way, to mix different graphics modes on the same display, using a few tricky PEEKs and POKEs.) Then you can actually get down to the business of drawing on the screen. A variety of methods is available for displaying graphics, the most common of which are to PLOT a point and to DRAWTO another point. PLOT is the equivalent of the TRS-80®'s SET/RESET command pair, and of the Apple's PLOT or HPLOT. DRAWTO corresponds to the Apple's HPLOT TO.

The following example of coding for the ATARI®should help to pull these various statements together.

```
400 GRAPHICS 4
410 SETCOLOR 4,3,5
420 SETCOLOR 0,9,8
430 COLOR 1
440 PLOT 0,20
450 DRAWTO 79,20
460 COLOR 0
470 PLOT 10,20
480 PLOT 20,20
490 PLOT 40,20
```

Line 400 initializes graphics mode 4, which has a resolution of 40 rows of 80 pixels on the upper five-sixths of the screen, plus four normal text lines at the bottom. The SETCOLOR statement in line 410 assigns a red-orange color (3) of luminance level 5 to color register 4, which becomes the background color. The second SETCOLOR statement assigns a light blue color (9) of luminance level 8 to color register 0. Henceforth, these two color/luminance combinations can be invoked using COLOR 0 and COLOR 1 respectively (according to the aforementioned chart in the *BASIC Manual*). The COLOR 1 statement in line 430, then, calls for the use of the data in color register 0 until further notice. The PLOT in line 440 changes the first pixel in the 20th line to this color and luminance, and the DRAWTO extends a horizontal line from that point to the right edge of the screen. Finally, the COLOR is changed back to that stored in register 4 (the background color) and three breaks are PLOTted in the line. (This pattern should sound familiar by now.)

Incidentally, one doesn't have to specify particular color/luminance combinations to be stored in the color registers. The ATARI® automatically sets up default values and, if you're content with these, you can forget about the SETCOLOR statement. But then, why would you want an ATARI® if you weren't going to exploit all its graphics versatility?

The extent of that versatility becomes apparent only when you realize that the ATARI® has a second built-in system for graphics display, known as Player/Missile graphics. In addition to the kinds of plotting and drawing already outlined, this system enables you to draw and, most significantly, to move graphics shapes on the screen with comparative ease. Up to four "players" and four smaller "missiles" (or five players and no missiles) can be defined and then manipulated around the screen. These can be displayed using as many as five color/luminance combinations of your choice, in addition to those already being used on the screen. The fatness and tallness of the shapes can be altered. And, you can specify different combinations of player/background priority, so that the moving shapes can seem either to disappear behind other graphics features or to move in front of them.

The mechanics of doing this are well beyond the scope of this article, and well offshore into that wine-dark sea that I mentioned at the beginning. But Player/Missile graphics programming is no more complex than many other types, and can generate spectacular results with relatively little coding. (Editor's note: See Sheldon Leemon's *Take-Apart Outer Space Attack*, elsewhere in this issue, for more on P/M graphics.)

So much for this eleventh-hour installment. Remember, there's only one month remaining in the "Guess the Identity of the Sensuous Programmer" contest, so send your entries NOW. No purchase is necessary, and you can enter as many times as you like. ***SoftSide*** employees and their families are not eligible. In the event of multiple correct entries, the winner will be chosen using the ***SoftSide*** Random Number Generator: spreading them all out on the floor and saying "eenie, meenie, miney, moe . . . "

Figure 1: Resolution and number of colors available in ATARI® graphics modes.

Graphics Mode	No. of rows (full screen)	Pixels per row	Total Pixels (full screen)	Colors available
3	24	40	960	4
4	48	80	3840	2
5	48	80	3840	4
6	96	160	15360	2
7	96	160	15360	4
8	192	320	61440	½*

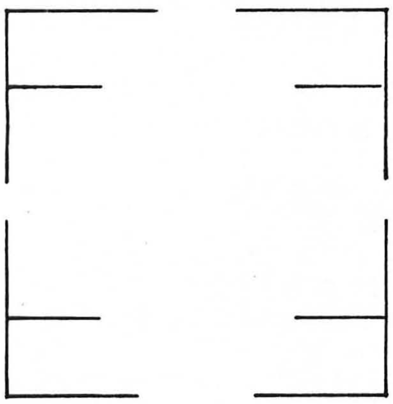
*Only one color may be used, but two different luminance levels may be chosen for the necessary contrast between background and drawing color.

Inkblot Patterns

By Joan Truckenbrod

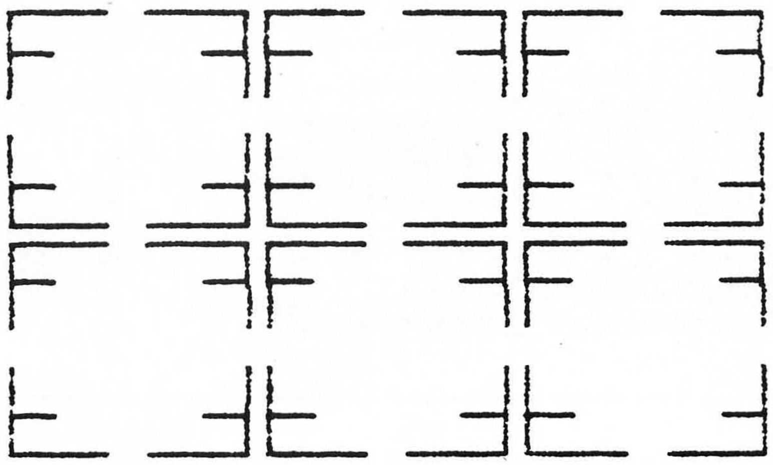
Horizontal and vertical mirror reflections can be used simultaneously in creating patterns. Both types of reflections have been illustrated separately in previous issues of *SoftSide*. Combining both horizontal and vertical mirror images in one pattern creates interesting shapes and figures.

This patterning system is similar to creating inkblot figures by putting a generous amount of ink or paint in the upper left-hand corner of a piece of paper, folding the paper (ink side in) in half, opening it, and then folding it in half in the other direction. In the inkblot that is created there are four figures, each a mirror reflection of the figure beside it and the figure above or below it. Using the letter "F" to represent these reflected figures, we can clarify the set of mirrored shapes.



The original figure is placed in the upper left-hand corner and is mirrored horizontally to the right, and vertically down to the lower left-hand corner. The mirror image of either of these reflected figures is then placed in the lower right-hand corner. This set of reflected figures includes the original letter "F", a backwards F, an upside-down F, and finally an F that is both upside-down and backwards.

A pattern naturally follows from the original set of four reflected figures as the reflecting process can continue in any direction. In a large pattern, such as the one at the top of this page, each figure is a reflection or mirror image of the neighboring figures.



In analyzing this pattern we see that the top row of figures consists of the original figure alternating with a figure that is the horizontal reflection of the original figure. The second row consists of the vertical reflection of the original figure alternating with a figure that has been reflected both horizontally and vertically. Consequently, there are four different versions of the figure to be drawn on the video screen.

Program Structure

The X and Y coordinates (in the range of 0 to 23) that define the original figure are listed in the DATA statement in line 95 of the program that follows. These values are placed in the X and Y arrays. The X and Y coordinates defining the remaining three figures are generated by the program in subroutine 4000. The values for the horizontally reflected figure are placed in the X1 and Y1 arrays; for the vertically reflected figure, in the X2 and Y2 arrays; and for the combined reflection, in arrays X3 and Y3. The number of coordinate points used to define the figure is specified in line 60.

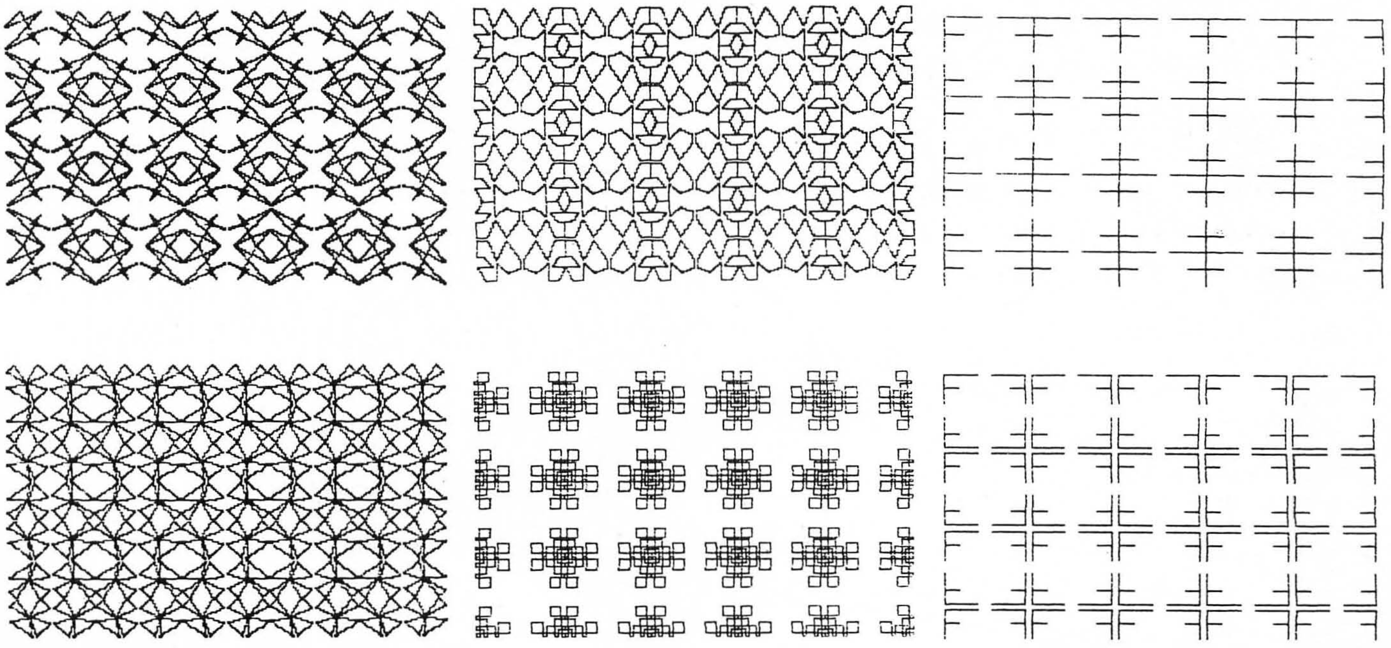
The program is structured to construct pairs of figures by using computed GOSUB statements. A counter variable L alternates in value between 1 and 2. If L is equal to 1 the first figure is drawn, and if L is equal to 2 the second figure is drawn. Based on the value of L, the program branches to the appropriate subroutine in which coordinate values are transferred to the NX and NY arrays.

The NX and NY arrays are used for drawing the figure on the screen. The drawing is performed in subroutine 3000. This subroutine also positions the figure in the correct position in the pattern.

In further analyzing the pattern, we see that horizontal rows 1, 3, 5 and 7 are the same, and rows 2, 4 and 6 are the same. Thus, the program is designed to create the pattern by generating pairs of rows. Lines 120-180 construct one row, and lines 190-240 construct the following row. The N1 FOR/NEXT loop repeats this pair of lines in order to construct the entire pattern.

This program can be used to create curved patterns and areas that are shaded in with color by carefully designing the original figure. Experiment with many different figures to create a wide variety of reflective or inkblot patterns.

Patterns of horizontal and vertical mirror reflection.

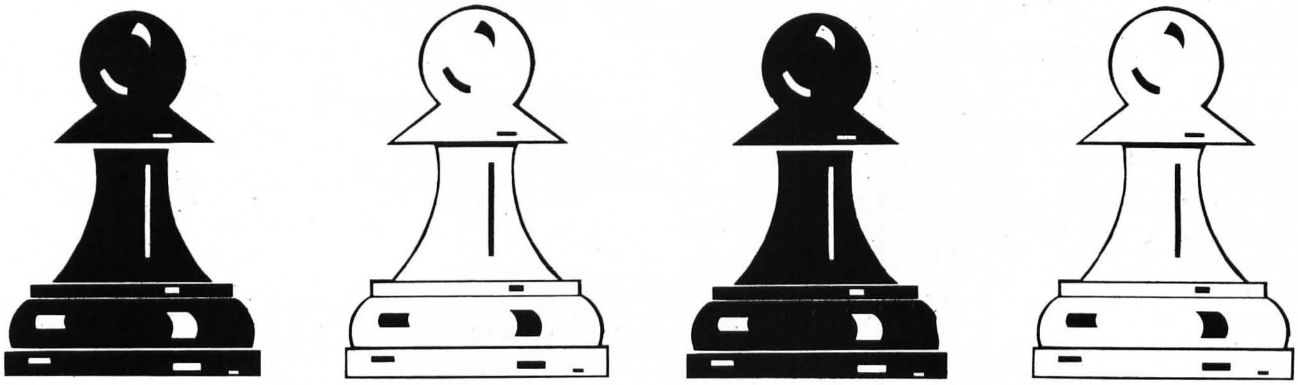


```
10  REM  HORIZONTAL AND VERTICAL
      REFLECTION
11  FOR N1 = 12 TO 160 STEP 48
50  DIM X(50),Y(50),X1(50),Y1(50)

53  DIM NX(50),NY(50)
55  DIM X2(50),Y2(50),X3(50),Y3(5
      0)
58  REM  NPTS IS THE NUMBER OF
      POINTS IN THE FIGURE
60 NPTS = 6
70  FOR I = 1 TO NPTS
80  READ X(I),Y(I)
90  NEXT I
95  DATA 20,2,2,2,2,10,10,10,2,10
      ,2,20
103  GOSUB 4000
104  HGR2
105  HCOLOR= 7
110  FOR N1 = 12 TO 160 STEP 48
115 L = 1
120  FOR M = 5 TO 240 STEP 24
140  ON L GOSUB 5000,5100
143 N = N1
145  GOSUB 3000
150 L = L + 1
160  IF L > 2 THEN L = 1
180  NEXT M
185 L = 1
190  FOR M = 5 TO 240 STEP 24
200  ON L GOSUB 5200,5300
210 L = L + 1
220  IF L > 2 THEN L = 1
223 N = N1 + 24
225  IF N1 > 160 GOTO 350
230  GOSUB 3000
240  NEXT M
250  NEXT N1
350  END
3000  REM  SCREEN PLOT ROUTINE
3005  IF NY(1) + N > 192 THEN  END
3010  HPLOT NX(1) + M,NY(1) + N
3020  FOR I = 2 TO NPTS
3030  HPLOT  TO NX(I) + M,NY(I) +
      N
3040  NEXT I
3050  RETURN
4000  REM
4010  REM  HORIZONTAL REFLECTION
4020  FOR I = 1 TO NPTS
4030 X1(I) = 24 - X(I)
4040 Y1(I) = Y(I)
4045  NEXT I
4050  REM  VERTICAL REFLECTION
4060  FOR I = 1 TO NPTS
4070 X2(I) = X(I)
4080 Y2(I) = 24 - Y(I)
4090  NEXT I
4100  REM HORIZONTAL AND VERTICAL
      REFLECTION (UPSIDE DOWN AND
      BACKWARDS)

4110  FOR I = 1 TO NPTS
4120 X3(I) = 24 - X(I)
4130 Y3(I) = 24 - Y(I)
4140  NEXT I
4150  RETURN
5000  FOR I = 1 TO NPTS
5010 NX(I) = X(I)
5020 NY(I) = Y(I)
5030  NEXT I
5040  RETURN
5100  FOR I = 1 TO NPTS
5110 NX(I) = X1(I)
5120 NY(I) = Y1(I)
5130  NEXT I
5140  RETURN
5200  FOR I = 1 TO NPTS
5210 NX(I) = X2(I)
5220 NY(I) = Y2(I)
5230  NEXT I
5240  RETURN
5300  FOR I = 1 TO NPTS
5310 NX(I) = X3(I)
5320 NY(I) = Y3(I)
5330  NEXT I
5340  RETURN
```

HEXAPAWN



THE MORE YOU PLAY, THE MORE YOU LOSE

by Carl Bevington
Translated by Alan J. Zett

***Hexapawn* is a simple marker game for a 16K TRS-80®, ATARI®or Apple (with Applesoft), in which the computer is programmed to "learn" from its previous mistakes.**

Martin Gardner devised the game of *Hexapawn* and its first version was published in *Scientific American* in 1962.

The marker game of *Hexapawn* is played on a three-by-three grid with the three bottom-row squares initially occupied by the player's pieces and the three upper-row squares initially occupied by the computer's pieces. The middle row of squares is vacant.

The player goes first, moving any of his markers in a forward (upward) direction. The computer's responding forward (downward) move will be chosen at random from a list of possible legal moves. The player and the computer alternate moves until a game is won.

As is true of the pawn in the game of *Chess*, only two legal moves are possible. First, a marker may be moved straight forward one square if the intended resting position is vacant. Second, a marker may be moved diagonally if the opponent occupies the intended resting position, in which case the opponent's marker is "captured" and removed from the playing board.

There are three ways in which to win a game:

1. By advancing a marker to the opponent's end of the playing board.

2. By capturing all of the opponent's markers. (This will seldom happen.)

3. By achieving a position that makes it impossible for the opponent to move any of his markers legally.

The program given here will enable the computer to "learn" heuristically. That is, it will be given a set of operating instructions which will enable it to improve its decision-making ability, based on the success or failure of previous decisions.

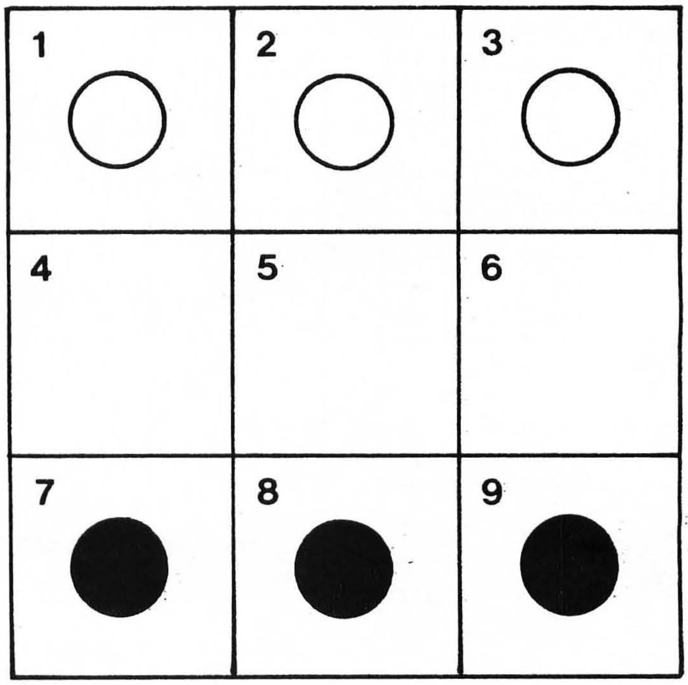


Figure 1

During the first few games, the computer wins occasionally by accident. Each time the player beats the computer, the computer's program will be altered so that it will not repeat the mistake that lost the game. Eventually, all possible errors are eliminated and the computer will win every game.

Table 1 gives a listing of the 37 possible marker positions on the playing board. Those numbered 1 through 3 are the possible board arrangements after the player's first move. Those numbered 4 through 23 are the possible board arrangements after the player's second move. Finally, those numbered 24 through 37 are the possible board arrangements after the player's third move.

In the column labeled Board Position Code, the digit 1 represents a player marker, the digit 3 represents a computer marker, and the digit 2 represents no marker. The nine consecutive digits describe the playing board from square number 1 through square number 9.

In the column labeled Possible Moves Code, the three possible computer moves are given by breaking the six-digit code into three two-digit groups, in which the first digit represents the computer's "from" move and the second digit represents the computer's "to" move.

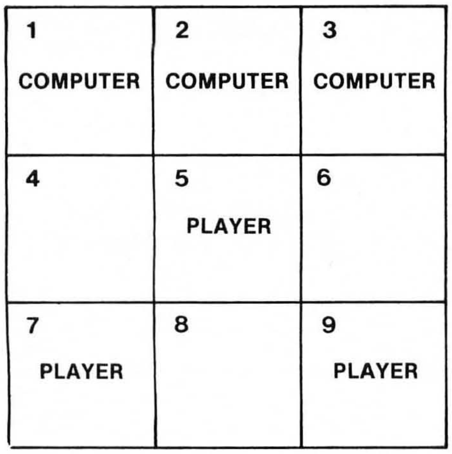


Figure 2

For example, if Figure 2 shows the position of the markers after the player's first move, then row 2 from Table 1 describes the current marker situation of the playing board.

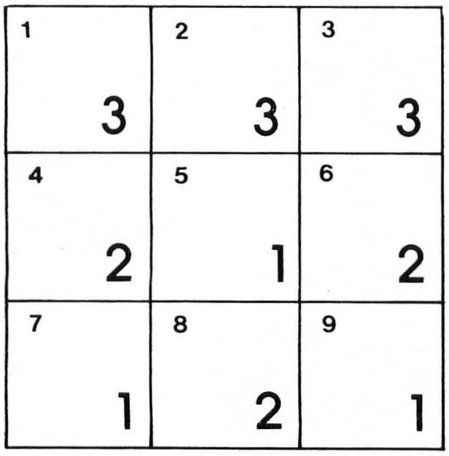


Figure 3

Using the previously described codes for the markers, Figure 3 represents the code characters for the playing board and 333212121 would be the Board Position Code. The three possible computer moves would be from 1 to 4, from 3 to 5, and from 3 to 6. Although a move from 1 to 5 would be legal, only three possible moves are coded.

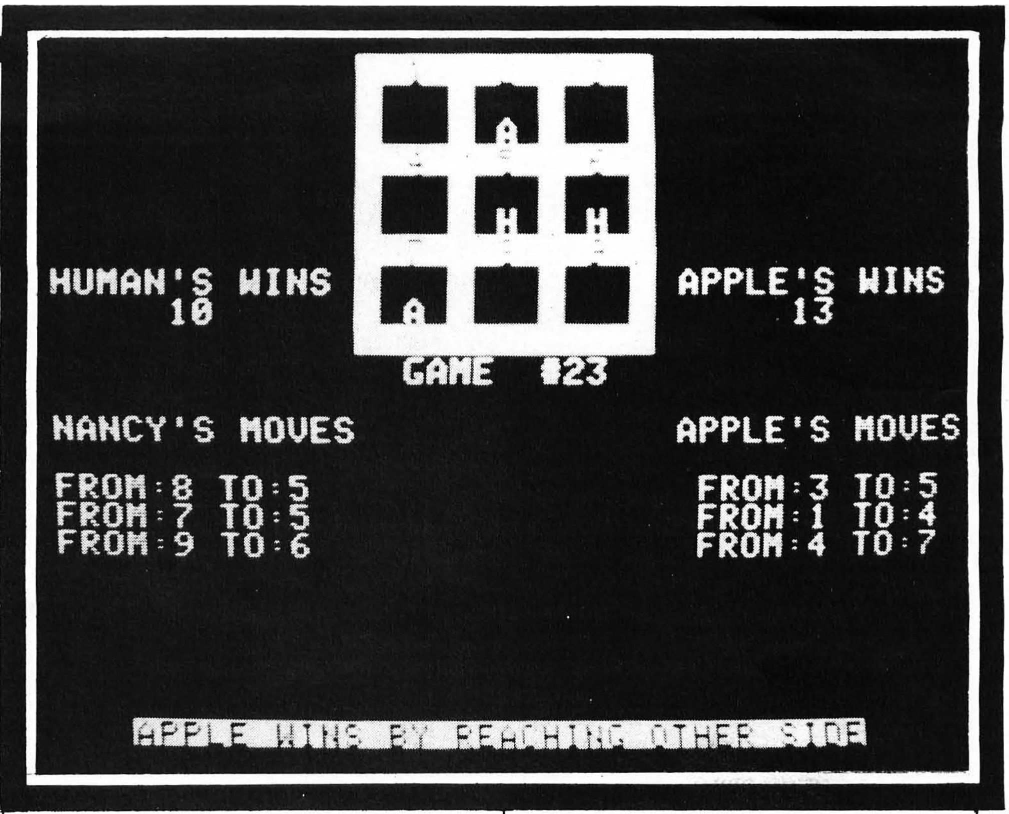
After the computer has randomly selected one of the three possible moves, the Possible Moves Code will remain unchanged if the move turns out to be favorable. If the move turns out to be unfavorable, the random move chosen will be changed to 99 in the Possible Moves Code. All moves coded 99 are rejected and another random move is chosen from the list in future games.

	BOARD POSITION CODE	POSSIBLE MOVES CODE
1	333122211	242536
2	333212121	143536
3	333221112	262514
4	233231122	265758
5	332132221	245958
6	323312221	154736
7	323213122	351469
8	323321212	474899
9	323123212	696899
10	233132221	243658
11	332231122	261458
12	233212221	353699
13	332212122	141599
14	332121221	242526
15	233121122	242526
16	233311122	263599
17	332113221	152499
18	323122221	369999
19	323221122	149999
20	233212122	353699
21	332212221	141599
22	323112212	153536
23	323211212	141535
24	322331222	475899
25	223133222	695899
26	232113222	246999
27	232311222	264799
28	223331222	475899
29	322133222	695899
30	232132222	245899
31	232231222	265899
32	223312222	473536
33	322213222	691514
34	322312222	154799
35	223213222	356999
36	223111222	359999
37	322111222	159999

Table 1

Variables

A#: Read in variable for code of possible board marker positions.
B$: INKEY$ for timing loop.
C$: Code for board positions after player's first move.
CF: Number of position for computer's move from.
CT: Number of position for computer's move to.
CW: Computer wins counter.
C(1) to C(9): Occupation code for nine marker positions.
D$: Code for board positions after player's second move.
D(1) to D(3): Possible random moves for computer.
E$: Code for board positions after player's third move.
F: Number of position for player's move from.
G: Random number move selected for computer's move.
I, J, K: Index counters for loops.
K$: Name of game.
M$(1) to M$(37): Codes for possible board marker positions.
N(1) to N(37): Codes for possible computer moves based on marker positions.
N$: Name of player.
P(1) to P(9): "Print at" positions for markers.
PW: Player wins counter.
Q: Index counter for delay loops.
T: Number of position for player's move to.
V: Number of stored code of board positions which matches current board.
W: Random number selected for computer move.
X, Y: Coordinates for setting graphics.
Y$: Name of Martin Gardner.
Z: Number of game counter.



```
HUMAN'S WINS                              APPLE'S WINS
     10                                        13
                     GAME  #23

NANCY'S MOVES                            APPLE'S MOVES

FROM:8  TO:5                              FROM:3  TO:5
FROM:7  TO:5                              FROM:1  TO:4
FROM:9  TO:6                              FROM:4  TO:7

          APPLE WINS BY REACHING OTHER SIDE
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      APPLESOFT BASIC          $
$         'HEXAPAWN'            $
$ AUTHOR: CARL A. BEVINGTON     $
$    TRANSL: ALAN J. ZETT       $
$    (C) 1982    SOFTSIDE       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization.

```
140  DIM M$(40),N(40),P(9,1): HOME
     :Y$ = "BY MARTIN GARDNER":K$
      = "H E X A P A W N"
```

Opening graphics cover and text.

```
180  INVERSE : FOR X = 1 TO 39 STEP
     2: FOR Y = 3 TO 23 STEP 10: HTAB
     X: VTAB Y: PRINT " ";: NEXT
     : NEXT : FOR X = 1 TO 39 STEP
     38: FOR Y = 3 TO 23 STEP 2: HTAB
     X: VTAB Y: PRINT " ";: NEXT
     : NEXT : NORMAL : HTAB 10: VTAB
     16: PRINT "PROGRAM CREATED B
     Y --";: HTAB 12: VTAB 18: PRINT
     "CARL A. BEVINGTON"
190  HTAB 15: PRINT "SALEM, OHIO"
     : HTAB 13: VTAB 6: FLASH : PRINT
     K$: HTAB 12: VTAB 8: NORMAL
     : PRINT Y$: HTAB 12: VTAB 21
     : PRINT "TRANSLATED BY AJZ":
      FOR X = 1 TO 5000: NEXT : HOME
```

Graphics for instructions.

```
210 AZ = 0: FLASH : FOR Y = 3 TO
     9 STEP 6: FOR X = 16 TO 24 STEP
     4: HTAB X: VTAB Y: PRINT  CHR$
     (65 + (7 * AZ));: NEXT :AZ =
     AZ + 1: NEXT
220  INVERSE : FOR Y = 1 TO 10 STEP
     3: HTAB 14: VTAB Y: PRINT  SPC(
     13): NEXT : FOR X = 14 TO 27
      STEP 4: FOR Y = 2 TO 9: HTAB
     X: VTAB Y: PRINT " ";: NEXT
     : NEXT
230 AZ = 0: FOR Y = 1 TO 7 STEP 3
     : FOR X = 16 TO 24 STEP 4:AZ
      = AZ + 1: HTAB X: VTAB Y: PRINT
     AZ;: NEXT : NEXT
```

Instructions for playing Hexapawn.

```
270  NORMAL : HTAB 1: VTAB 12: CALL
      - 958: PRINT "THE GAME OF H
     EXAPAWN IS PLAYED ON A 3X3 G
     RID WITH THREE MARKERS FOR E
     ACH PLAYER.THE MARKERS ARE M
     OVED IN THE SAME MANNERAS PA
     WNS IN CHESS:": PRINT : PRINT
     "1. A MARKER MAY BE MOVED FO
     RWARD ONE"
280  PRINT "   CELL TO AN EMPTY C
     ELL.": PRINT "2. A MARKER MA
     Y BE MOVED FORWARD": PRINT "
         DIAGONALLY TO THE LEFT OR
       RIGHT TO      CAPTURE AN OP
      PONENT'S MARKER. THE       C
      APTURED MARKER IS TAKEN OUT
      OF PLAY."
```

Read codes for possible board positions, random moves, and print positions for markers.

```
320 Z = 1: FOR I = 1 TO 37: READ
     A:M$(I) =  STR$ (A): READ N(
     I): NEXT I: FOR I = 1 TO 9: READ
     P(I,0),P(I,1): NEXT I
330  PRINT  SPC( 5): INVERSE : PRINT
     " PRESS SPACE BAR TO CONTINU
     E";: NORMAL : POKE  - 16368,
     0: CALL  - 756
```

More instructions for play.

```
370  HTAB 1: VTAB 12: CALL  - 958
     : PRINT "THE GAME IS WON IN
     THREE POSSIBLE WAYS: ": PRINT
     "1. BY MOVING A MARKER TO TH
     E OPPONENT'S    SIDE OF THE
     GRID.": PRINT "2. BY CAPTURI
     NG ALL OF THE OPPONENT'S
       MARKERS."
380  PRINT "3. BY PLANNING MOVES
     SO THAT A POINT IS    REACHE
     D IN THE GAME WHEN THE OPPON
     ENT   IS UNABLE TO MOVE ANY
     MARKERS.": PRINT
390  PRINT  SPC( 5): INVERSE : PRINT
     " PRESS SPACE BAR TO CONTINU
     E";: NORMAL : CALL  - 756
400  HTAB 1: VTAB 12: CALL  - 958
     : PRINT "ALTHOUGH THE COMPUT
     ER IS PROGRAMMED TO  MAKE IT
     S DIFFERENT MOVES AT RANDOM,
      THE COMPUTER WILL 'REMEMBER
     ' HOW IT LOST A  GAME AND WI
     LL NOT TRY THE SAME RANDOM
      MOVE AGAIN.": PRINT : PRINT
410  INPUT "ENTER YOUR FIRST NAME
      ==>";N$
```

Graphics for playing board.

```
450  HTAB 1: VTAB 11: CALL  - 958
     : GOSUB 2410
460  HTAB 1: VTAB 8: PRINT "HUMAN
     'S WINS";: HTAB 28: PRINT "A
     PPLE'S WINS";: GOSUB 2410:C$
      = "":D$ = "":E$ = "": FOR J
      = 1 TO 7 STEP 3: FOR I = J TO
     J + 2:C(I) = 4 -  INT ((I +
     2) / 3): NEXT : NEXT :V = 0:
     W = 0:BN = 0:MN = 0
```

First player move entered here.

```
500  HTAB 1: VTAB 15: PRINT  SPC(
     40): HTAB 1: VTAB 15: INPUT
     "FROM:";F: HTAB 8: VTAB 15: INPUT
     "TO:";T
```

Check for valid player move.

```
540  IF F > 9 OR T > 9 THEN 500
550  IF C(F) <  > 1 THEN 500
560  IF C(T) <  > 2 THEN 500
570  IF F - T <  > 3 THEN 500
```

Set markers on playing board.

```
610  GOSUB 2140
620 C(F) = 2:C(T) = 1
```

Find code for current playing board marker positions.

```
660  FOR I = 1 TO 9:C$ =  STR$ (1
     0 *  VAL (C$) + C(I)): NEXT
     I
670  FOR I = 1 TO 3
680  IF C$ = M$(I) THEN V = I
690  NEXT
```

Determine one of three possible random computer moves.

```
730 D(1) =  INT (N(V) / 10000):D =
     N(V) - 10000 * D(1)
740 D(2) =  INT (D / 100)
750 D(3) = D - 100 * D(2)
760 G =  INT ( RND (1) * 3) + 1
770  IF D(G) = 99 THEN 760
780 CF =  INT (D(G) / 10):CT = D(
     G) - 10 * CF:W = G
```

Print first computer move.

```
820  HTAB 29: VTAB 15: PRINT "FRO
     M:";CF;: HTAB 36: PRINT "TO:
     ";CT;
830  FOR Q = 1 TO 1200: NEXT Q
840 C(CF) = 2:C(CT) = 3
850  GOSUB 2180
```

Second player move entered here.

```
890  HTAB 1: VTAB 16: PRINT  SPC(
     40): HTAB 1: VTAB 16: INPUT
     "FROM:";F: HTAB 8: VTAB 16: INPUT
     "TO:";T
```

Check for valid player move.

```
930  IF F > 9 OR T > 9 THEN 890
940  IF C(F) <  > 1 THEN 890
950  IF F - T = 3 AND C(T) = 2 THEN
     1020
960  IF F - T = 2 AND C(T) = 3 THEN
     1020
970  IF F - T = 4 AND C(T) = 3 THEN
     1020
980  GOTO 890
```

Set marker on playing board.

```
1020 C(F) = 2:C(T) = 1: GOSUB 214
     0
1030  IF T < 4 THEN 2220
1040  IF C(1) * C(3) * C(5) = 27 THEN
     2310
```

Find code for current playing board marker positions.

```
1080  FOR I = 1 TO 9:D$ =  STR$ (
     10 *  VAL (D$) + C(I)): NEXT

1090 BN = V:MN = W: FOR I = 4 TO
     23
1100  IF D$ = M$(I) THEN V = I
1110  NEXT
```

Determine one of three possible random computer moves.

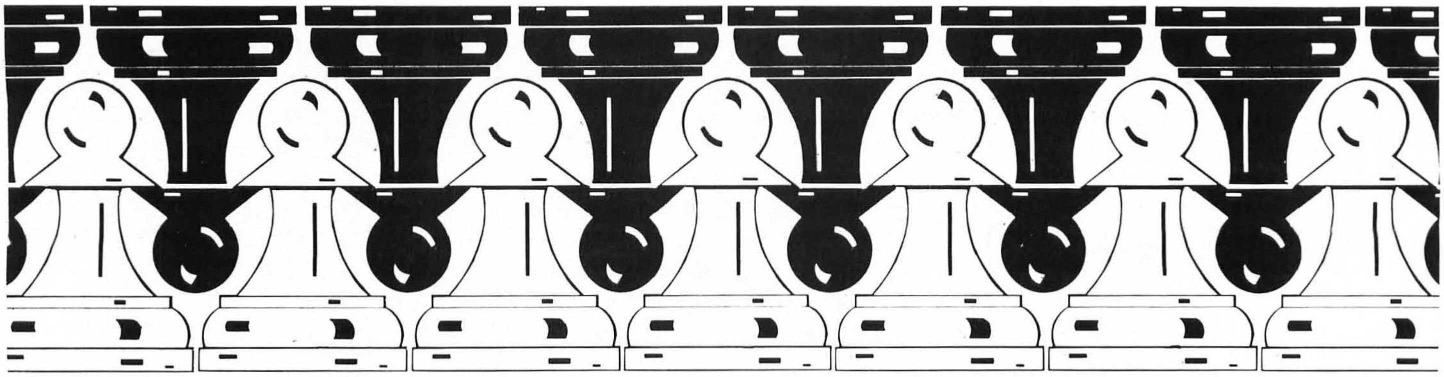
```
1150 D(1) =  INT (N(V) / 10000):D
     = N(V) - 10000 * D(1)
1160 D(2) =  INT (D / 100)
1170 D(3) = D - 100 * D(2)
1180 G =  INT ( RND (1) * 3) + 1
1190  IF D(G) = 99 THEN 1180
1200 CF =  INT (D(G) / 10):CT = D
     (G) - 10 * CF:W = G
```

Print second computer move.

```
1240  HTAB 29: VTAB 16: PRINT "FR
     OM:";CF;: HTAB 36: PRINT "TO
     :";CT;
1250  FOR Q = 1 TO 1200: NEXT Q
1260 C(CF) = 2:C(CT) = 3:W = G
1270  GOSUB 2180
```

Check for possible computer win.

```
1310  IF CT > 6 THEN 2360
1320  IF V = 12 AND D(G) = 36 THEN
     1400
1330  IF V = 13 AND D(G) = 14 THEN
     1400
1340  IF V = 14 AND D(G) = 26 THEN
     1400
1350  IF V = 15 AND D(G) = 24 THEN
     1400
1360  IF V = 22 AND D(G) = 35 THEN
     1400
1370  IF V = 23 AND D(G) = 15 THEN
     1400
1380  IF V = 18 OR V = 19 THEN 14
     00
1390  GOTO 1450
1400 CW = CW + 1: HTAB 33: VTAB 9
     : PRINT CW;: HTAB 10: VTAB 2
     4: INVERSE : PRINT "APPLE WI
     NS BY BLOCKING";: NORMAL : GOTO
     2420
```



```
1410  GOTO 460
```

Third player move entered here.

```
1450  HTAB 1: VTAB 17: PRINT  SPC(
     40): HTAB 1: VTAB 17: INPUT
     "FROM:";F: HTAB 8: VTAB 17: INPUT
     "TO:";T
```

Check for valid player move.

```
1490  IF F > 9 OR T > 9 THEN 1450

1500  IF C(F) <  > 1 THEN 1450
1510  IF F - T = 3 AND C(T) = 2 THEN
     1550
1520  IF F - T = 2 AND C(T) = 3 THEN
     1550
1530  IF F - T = 4 AND C(T) = 3 THEN
     1550
1540  GOTO 1450
1550 C(F) = 2:C(T) = 1
1560  IF (C(2) * C(5) = 3) AND (C
     (1) * C(3) * C(4) * C(6) * C
     (7) * C(8) * C(9) = 128) THEN
     2310
```

Set marker on playing board.

```
1600  GOSUB 2140
1610  IF T < 4 THEN 2220
```

Find code for current playing board marker positions.

```
1650  FOR I = 1 TO 9:E$ =  STR$ (
     10 *  VAL (E$) + C(I)): NEXT

1660 BN = V:MN = W: FOR I = 24 TO
     37
1670  IF E$ = M$(I) THEN V = I
1680  NEXT
```

Determine one of three possible random computer moves.

```
1720 D(1) =  INT (N(V) / 10000):D
      = N(V) - 10000 * D(1)
1730 D(2) =  INT (D / 100)
1740 D(3) = D - 100 * D(2)
1750 G =  INT ( RND (1) * 3) + 1
1760  IF D(G) = 99 THEN 1750
1770 CF =  INT (D(G) / 10):CT = D
     (G) - 10 * CF:W = G
```

Print third computer move.

```
1810  HTAB 29: VTAB 17: PRINT "FR
     OM:";CF;: HTAB 36: PRINT "TO
     :";CT;
1820  FOR Q = 1 TO 1200: NEXT Q
1830 C(CF) = 2:C(CT) = 3:W = G
1840  GOSUB 2180
```

Check for possible computer win.

```
1880  IF CT > 6 THEN 2360
1890  IF V = 34 AND D(G) = 15 THEN
     2370
1900  IF V = 35 AND D(G) = 35 THEN
     2370
1910  IF V = 30 AND D(G) = 24 THEN
     2370
1920  IF V = 31 AND D(G) = 26 THEN
     2370
1930  IF V = 32 AND D(G) = 35 THEN
     2370
1940  IF V = 33 AND D(G) = 15 THEN
     2370
```

Fourth player move entered here.

```
1980  HTAB 1: VTAB 18: PRINT  SPC(
     40): HTAB 1: VTAB 18: INPUT
     "FROM:";F: HTAB 8: VTAB 18: INPUT
     "TO:";T
```

Check for valid player move.

```
2020  IF F > 9 OR T > 9 THEN 1980
2030  IF C(F) <  > 1 THEN 1980
2040  IF C(T) <  > 2 THEN 1980
2050  IF F - T <  > 3 THEN  THEN
     1980
```

Set marker on playing board.

```
2090  GOSUB 2140
2100  GOTO 2220
```

Graphics subroutine for player move.

```
2140  HTAB P(F,0): VTAB P(F,1): PRINT
     " ";: HTAB P(T,0): VTAB P(T,
     1): PRINT "H";: RETURN
```

Graphics subroutine for computer move.

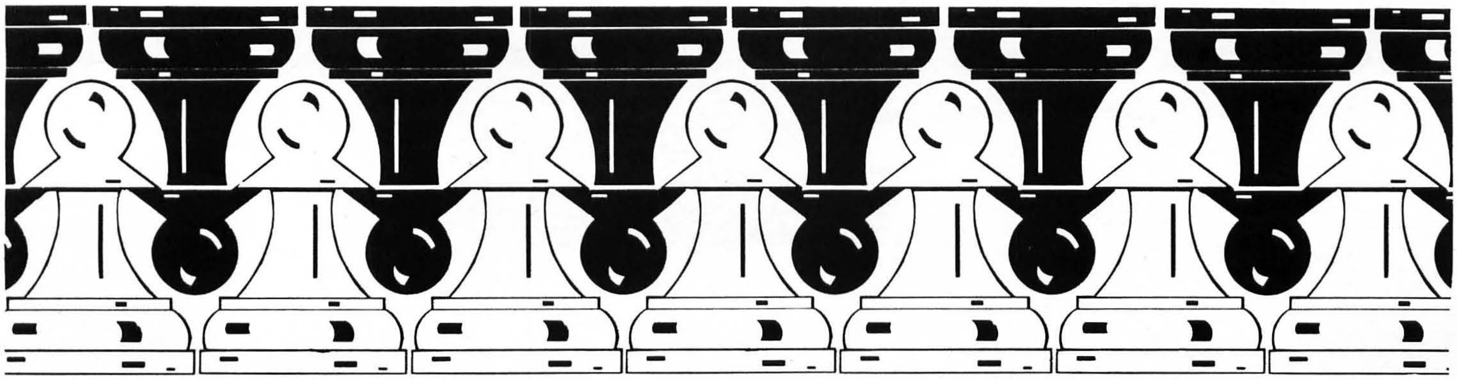
```
2180  HTAB P(CF,0): VTAB P(CF,1):
     PRINT " ";: HTAB P(CT,0): VTAB
     P(CT,1): PRINT "A";: RETURN
```

Player win announcement.

```
2220 PW = PW + 1: HTAB 6: VTAB 9:
     PRINT PW;: HTAB 4: VTAB 24:
     INVERSE : PRINT "HUMAN WINS
     BY REACHING OTHER SIDE";: NORMAL
```

Reset possible random moves for computer.

```
2260 D(W) = 99
2270  IF W = 1 THEN N(V) = 990000
     + 100 * D(2) + D(3)
2280  IF W = 2 THEN N(V) = 10000 *
     D(1) + 9900 + D(3)
2290  IF W = 3 THEN N(V) = 10000 *
     D(1) + 100 * D(2) + 99
2300  IF D(1) + D(2) + D(3) = 297
     THEN 2600
2305  GOTO 2420
```



```
2310  GOSUB 2140
```

Player/computer win announcements.

```
2350 PW = PW + 1: HTAB 6: VTAB 9:
      PRINT PW;: HTAB 10: VTAB 24
     : INVERSE : PRINT "HUMAN WIN
     S BY BLOCKING";: NORMAL : GOTO
     2260
2360 CW = CW + 1: HTAB 33: VTAB 9
     : PRINT CW;: HTAB 4: VTAB 24
     : INVERSE : PRINT "APPLE WIN
     S BY REACHING OTHER SIDE";::
      NORMAL : GOTO 2420
2370 CW = CW + 1: HTAB 33: VTAB 9
     : PRINT CW;: HTAB 2: VTAB 24
     : INVERSE : PRINT "APPLE WIN
     S BY TAKING HUMAN'S MARKERS"
     ;: NORMAL : GOTO 2420
```

Screen wipe for beginning of new game.

```
2410  FOR I = 1 TO 3: HTAB P(I,0)
     : VTAB P(I,1): PRINT "A";: NEXT
     : FOR I = 4 TO 6: HTAB P(I,0
     ): VTAB P(I,1): PRINT " ";: NEXT
     : FOR I = 7 TO 9: HTAB P(I,0
     ): VTAB P(I,1): PRINT "H";: NEXT

2415  HTAB 16: VTAB 11: PRINT "GA
     ME  #";Z;: HTAB 1: VTAB 13: PRINT
     N$;"'S MOVES";: HTAB 28: PRINT
     "APPLE'S MOVES";: RETURN
2420 Z = Z + 1: FOR Q = 1 TO 5000
     : NEXT : HTAB 1: VTAB 14: CALL
      - 958: HTAB 23: VTAB 11: PRINT
     Z;: GOTO 460
```

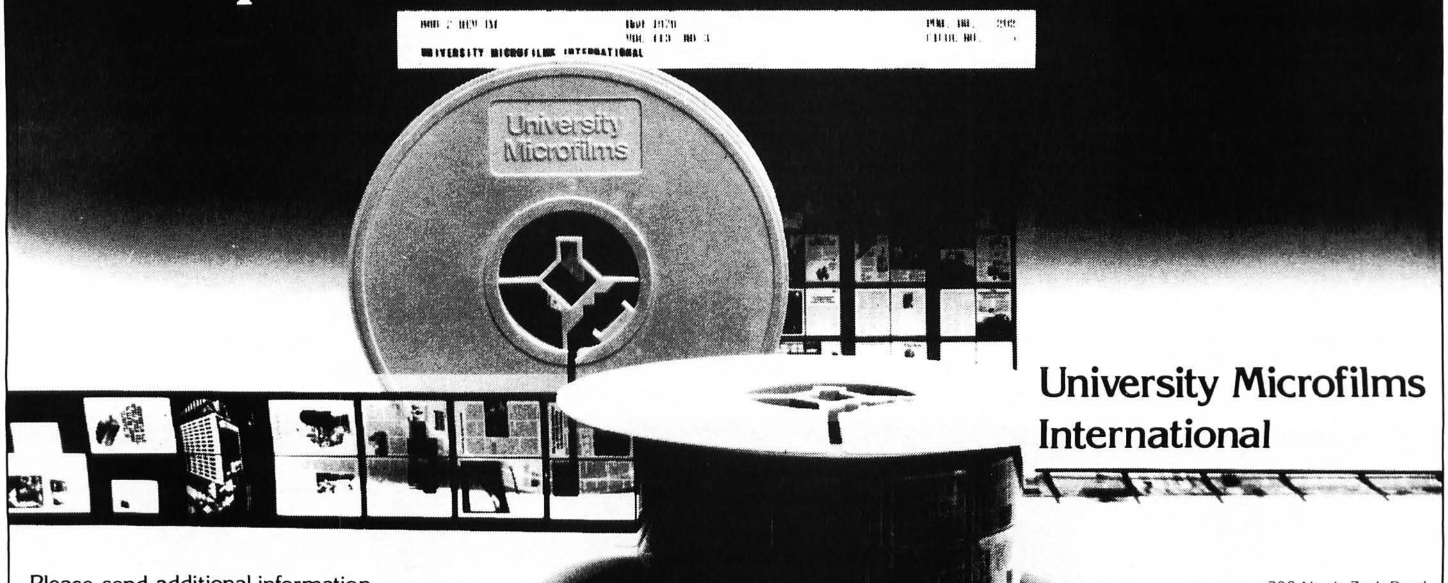
Data needed for possible board positions, random moves, and print positions for markers.

```
2460  DATA 333122211,242536,33321
     2121,143536,333221112,262514
     ,233231122,265758,332132221,
     245958,323312221,154736,3232
     13122,351469,323321212,47489
     9,323123212,696899,233132221
     ,243658,332231122,261458,233
     212221,353699,332212122,1415
     99,332121221
2470  DATA 242526,233121122,24252
     6,233311122,263599,332113221
     ,152499,323122221,369999,323
     221122,149999,233212122,3536
     99,332212221,141599,32311221
     2,153536,323211212,141535,32
     2331222,475899,223133222,695
     899,232113222,246999,2323112
     22,264799
2480  DATA 223331222,475899,32213
     3222,695899,232132222,245899
     ,232231222,265899,223312222,
     473536,322213222,691514,3223
     12222,471599,223213222,69359
     9,223111222,359999,322111222
     ,159999,16,3,20,3,24,3,16,6,
     20,6,24,6,16,9,20,9,24,9
```

Erase move on last board if all moves in current board lose.

```
2600 D(1) =  INT (N(BN) / 10000):
      D = N(BN) - 10000 * D(1)
2610 D(2) =  INT (D / 100)
2620 D(3) = D - 100 * D(2)
2630 D(MN) = 99
2640  IF MN = 1 THEN N(BN) = 9900
      00 + 100 * D(2) + D(3)
2650  IF MN = 2 THEN N(BN) = 1000
      0 * D(1) + 9900 + D(3)
2660  IF MN = 3 THEN N(BN) = 1000
      0 * D(1) + 100 * D(2) + 99
2670  GOTO 2420
```

This publication is available in microform.



University Microfilms International

Please send additional information
for______________________________
(name of publication)
Name_____________________________
Institution_______________________
Street____________________________
City______________________________
State__________ Zip________________

300 North Zeeb Road
Dept. P.R.
Ann Arbor, Mi. 48106
U.S.A.

30-32 Mortimer Street
Dept. P.R.
London WIN 7RA
England

```
    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
    $         ATARI  BASIC         $
    $          "HEXAPAWN"          $
    $ AUTHOR: CARL A. BEVINGTON    $
    $    TRANSL: ALAN J. ZETT      $
    $    (C) 1982    SOFTSIDE      $
    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization.

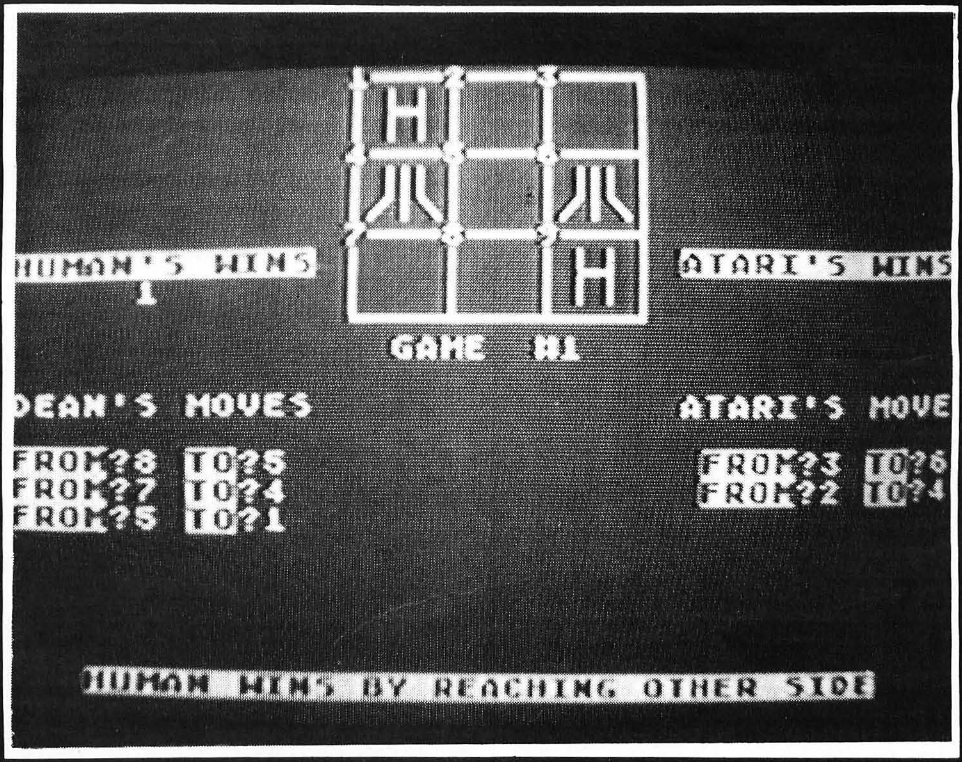
```
140 CLR :DIM M$(342),N(40),P(9,2),Y$(1
7),K$(15),N$(9),C$(9),D$(9),E$(9),C(9)
,D(3):M$(342)="#"
150 GRAPHICS 0:Y$="By Martin Gardner":
K$="H E X A P A W N":POKE 752,1
160 REM
162 REM LINES 180-186:
164 REM 'b'='CTRL B' : 'f'='CTRL F'
        'g'='CTRL G' : 'n'='CTRL N'
        'v'='CTRL V' : 'i'='SHIFT ='
166 REM 'l'='ESC CTRL +'
        'd'='ESC CTRL ='
168 REM
```

Opening graphics cover and text.

```
180 FOR X=0 TO 38 STEP 2:FOR Y=2 TO 22
 STEP 10:POSITION X,Y:? "_";:NEXT Y:NE
XT X:DIM A$(10):A$="bivllldfig"
183 FOR X=0 TO 38 STEP 38:FOR Y=2 TO 2
2 STEP 2:POSITION X,Y:? "_";:NEXT Y:NE
XT X:DIM H$(10):H$="bnvllldb v"
186 POSITION 9,15:? "PROGRAM CREATED B
Y --";:POSITION 11,17:? "Carl A. Bevin
gton":DIM BL$(10):BL$="   llld    "
190 POSITION 14,18:? "SALEM, OHIO":POS
ITION 12,5:? K$:POSITION 11,7:? Y$:POS
ITION 11,20:? "TRANSLATED BY AJZ"
195 FOR X=1 TO 1600:NEXT X:GRAPHICS 0:
POKE 752,1:POKE 82,0
200 REM
202 REM IN LINES 210-219:
204 REM 'c'='CTRL C' : 'd'='CTRL D'
        'e'='CTRL E' : 'r'='CTRL R'
        'x'='CTRL X'
206 REM 'z'='CTRL C' : '1-9'='1-9'
        'i'='SHIFT ='
208 REM
```

Graphics for instructions.

```
210 POSITION 13,0:? "1rrr2rrr3rrre"
211 POSITION 13,1:? "i   i   i   i"
212 POSITION 13,2:? "i   i   i   i"
213 POSITION 13,3:? "4rrr5rrr6rrrd"
214 POSITION 13,4:? "i   i   i   i"
215 POSITION 13,5:? "i   i   i   i"
216 POSITION 13,6:? "7rrr8rrr9rrrd"
```



```
217 POSITION 13,7:? "i   i   i   i"
218 POSITION 13,8:? "i   i   i   i"
219 POSITION 13,9:? "zrrrxrrrxrrrc"
220 FOR X=14 TO 22 STEP 4:POSITION X,1
:? A$;:POSITION X,7:? H$;:NEXT X:CLS=2
500
```

Instructions for playing Hexapawn.

```
270 POS=11:GOSUB CLS:? "The game of HE
XAPAWN is played on a 3X3 grid with th
ree markers for each player.";
275 ? "The markers are moved in the sa
me manneras chess:":? :? "1. A marker
may be moved forward one":OPEN #1,4,0,
"K"
280 ? "   cell to an empty cell.":? "2
. A marker may be moved forward":? "
diagonally to the left or right to"
285 ? "   capture an opponent's marker
. The":? "   captured marker is taken
out of play.":RESTORE
```

Read codes for possible board positions, random moves, and print positions for markers.

```
320 Z=1:FOR I=1 TO 37:READ A:M$(I*9,I*
9+8)=STR$(A):READ AZ:N(I)=AZ:NEXT I:FO
R I=1 TO 9:READ AZ:P(I,0)=AZ
330 READ AZ:P(I,1)=AZ:NEXT I:? "
PRESS SPACE BAR TO CONTINUE ";:POKE 76
4,255:GET #1,AZ
```

More instructions for play.

```
370 POS=11:GOSUB CLS:? "The game is wo
n in three possible ways: ":? "1. By m
oving a marker to the opponents"
375 ? "   side of the grid.":? "2. By
capturing all of the opponent's      m
arkers."
380 ? "3. By planning moves so that a
point is    reached in the game when t
he opponent   is unable to move any ma
rkers."
390 ? :? "        PRESS SPACE BAR TO CON
TINUE ";:GET #1,AZ
400 POS=11:GOSUB CLS:? "Although the c
omputer is programmed to  make its dif
ferent moves at random,"
405 ? "the computer will 'remember' ho
w it losta game and will not try the s
ame random move again.":? :?
410 POKE 752,0:? "Enter your first nam
e ";:INPUT N$
```

Graphics for playing board.

```
450 POKE 752,1:POS=10:GOSUB CLS:GOSUB
2410
460 POSITION 0,7:? "HUMAN'S WINS";:POS
ITION 27,7:? "ATARI'S WINS";:GOSUB 241
0:C$="0":D$="0":E$="0"
470 FOR J=1 TO 7 STEP 3:FOR I=J TO J+2
:C(I)=4-INT((I+2)/3):NEXT I:NEXT J:V=0
:W=0:BN=0:MN=0
```

First player move entered here.

```
500 POS=14:GOSUB CLS:? "FROM";:INPUT F
:POSITION 7,14:? "TO";:INPUT T
```

Check for valid player move.

```
540 IF F>9 OR T>9 THEN 500
550 IF C(F)<>1 THEN 500
560 IF C(T)<>2 THEN 500
570 IF F-T<>3 THEN 500
```

Set markers on playing board.

```
610 GOSUB 2140
620 C(F)=2:C(T)=1
```

Find code for current playing board marker positions.

```
660 FOR I=1 TO 9:C$=STR$(10*VAL(C$)+C(
I)):NEXT I
670 FOR I=1 TO 3
680 IF C$=M$(I*9,I*9+8) THEN V=I
690 NEXT I
```

Determine one of three possible random computer moves.

```
730 D(1)=INT(N(V)/10000):D=N(V)-10000*
D(1)
740 D(2)=INT(D/100)
750 D(3)=D-100*D(2)
760 G=INT(RND(1)*3)+1
770 IF D(G)=99 THEN 760
780 CF=INT(D(G)/10):CT=D(G)-10*CF:W=G
```

Print first computer move.

```
820 POSITION 28,14:? "FROM?";CF;" TO?"
;CT;
830 FOR Q=1 TO 400:NEXT Q
840 C(CF)=2:C(CT)=3
850 GOSUB 2180
```

Second player move entered here.

```
890 POS=15:GOSUB CLS:? "FROM";:INPUT F
:POSITION 7,15:? "TO";:INPUT T
```

Check for valid player move.

```
930 IF F>9 OR T>9 THEN 890
940 IF C(F)<>1 THEN 890
950 IF F-T=3 AND C(T)=2 THEN 1020
960 IF F-T=2 AND C(T)=3 THEN 1020
970 IF F-T=4 AND C(T)=3 THEN 1020
980 GOTO 890
```

Set marker on playing board.

```
1020 C(F)=2:C(T)=1:GOSUB 2140
1030 IF T<4 THEN 2220
1040 IF C(1)*C(3)*C(5)=27 THEN 2310
```

Find code for current playing board marker positions.

```
1080 FOR I=1 TO 9:D$=STR$(10*VAL(D$)+C
(I)):NEXT I
1090 BN=V:MN=W:FOR I=4 TO 23
1100 IF D$=M$(I*9,I*9+8) THEN V=I
1110 NEXT I
```

Determine one of three possible random computer moves.

```
1150 D(1)=INT(N(V)/10000):D=N(V)-10000
*D(1)
1160 D(2)=INT(D/100)
1170 D(3)=D-100*D(2)
1180 G=INT(RND(1)*3)+1
1190 IF D(G)=99 THEN 1180
1200 CF=INT(D(G)/10):CT=D(G)-10*CF:W=G
```

Print second computer move.

```
1240 POSITION 28,15:? "FROM?";CF;" TO?
";CT;
1250 FOR Q=1 TO 400:NEXT Q
1260 C(CF)=2:C(CT)=3:W=G
1270 GOSUB 2180
```

Check for possible computer win.

```
1310 IF CT>6 THEN 2360
1320 IF V=12 AND D(G)=36 THEN 1400
1330 IF V=13 AND D(G)=14 THEN 1400
1340 IF V=14 AND D(G)=26 THEN 1400
1350 IF V=15 AND D(G)=24 THEN 1400
1360 IF V=22 AND D(G)=35 THEN 1400
1370 IF V=23 AND D(G)=15 THEN 1400
1380 IF V=18 OR V=19 THEN 1400
1390 GOTO 1450
1400 CW=CW+1:POSITION 32,8:? CW;:POSIT
ION 9,22:? "ATARI WINS BY BLOCKING";:G
OTO 2420
1410 GOTO 460
```

Third player move entered here.

```
1450 POS=16:GOSUB CLS:? "FROM";:INPUT
F:POSITION 7,16:? "TO";:INPUT T
```

Check for valid player move.

```
1490 IF F>9 OR T>9 THEN 1450
1500 IF C(F)<>1 THEN 1450
1510 IF F-T=3 AND C(T)=2 THEN 1550
1520 IF F-T=2 AND C(T)=3 THEN 1550
1530 IF F-T=4 AND C(T)=3 THEN 1550
1540 GOTO 1450
1550 C(F)=2:C(T)=1
1560 IF (C(2)*C(5)=3) AND (C(1)*C(3)*C
(4)*C(6)*C(7)*C(8)*C(9)=128) THEN 2310
```

Set marker on playing board.

```
1600 GOSUB 2140
1610 IF T<4 THEN 2220
```

Find code for current playing board marker positions.

```
1650 FOR I=1 TO 9:E$=STR$(10*VAL(E$)+C
(I)):NEXT I
1660 BN=V:MN=W:FOR I=24 TO 37
1670 IF E$=M$(I*9,I*9+8) THEN V=I
1680 NEXT I
```

Determine one of three possible random computer moves.

```
1720 D(1)=INT(N(V)/10000):D=N(V)-10000
*D(1)
1730 D(2)=INT(D/100)
1740 D(3)=D-100*D(2)
1750 G=INT(RND(1)*3)+1
1760 IF D(G)=99 THEN 1750
1770 CF=INT(D(G)/10):CT=D(G)-10*CF:W=G
```

Print third computer move.

```
1810 POSITION 28,16:? "FROM?";CF;" TO?
";CT;
1820 FOR Q=1 TO 400:NEXT Q
1830 C(CF)=2:C(CT)=3:W=G
1840 GOSUB 2180
```

Check for possible computer win.

```
1880 IF CT>6 THEN 2360
1890 IF V=34 AND D(G)=15 THEN 2370
1900 IF V=35 AND D(G)=35 THEN 2370
1910 IF V=30 AND D(G)=24 THEN 2370
1920 IF V=31 AND D(G)=26 THEN 2370
1930 IF V=32 AND D(G)=35 THEN 2370
1940 IF V=33 AND D(G)=15 THEN 2370
```

Fourth player move entered here.

```
1980 POS=17:GOSUB CLS:? "FROM";:INPUT
F:POSITION 7,17:? "TO";:INPUT T
```

Check for valid player move.

```
2020 IF F>9 OR T>9 THEN 1980
2030 IF C(F)<>1 THEN 1980
2040 IF C(T)<>2 THEN 1980
2050 IF F-T<>3 THEN 1980
```

Set marker on playing board.

```
2090 GOSUB 2140
2100 GOTO 2220
```

Graphics subroutine for player move.

```
2140 POSITION P(F,0),P(F,1):? BL$;:POS
ITION P(T,0),P(T,1):? H$;:RETURN
```

Graphics subroutine for computer move.

```
2180 POSITION P(CF,0),P(CF,1):? BL$;:P
OSITION P(CT,0),P(CT,1):? A$;:RETURN
```

Player win announcement.

```
2220 PW=PW+1:POSITION 5,8:? PW;:POSITI
ON 3,22:? "HUMAN WINS BY REACHING OTHE
R SIDE";
```

Reset possible random moves for computer.

```
2260 D(W)=99
2270 IF W=1 THEN N(V)=990000+100*D(2)+
D(3)
2280 IF W=2 THEN N(V)=10000*D(1)+9900+
D(3)
2290 IF W=3 THEN N(V)=10000*D(1)+100*D
(2)+99
2300 IF D(1)+D(2)+D(3)=297 THEN 2600
2305 GOTO 2420
2310 GOSUB 2140
```

Player/computer win announcements.

```
2350 PW=PW+1:POSITION 5,8:? PW;:POSITI
ON 9,22:? "HUMAN WINS BY BLOCKING";:GO
TO 2260
2360 CW=CW+1:POSITION 32,8:? CW;:POSIT
ION 3,22:? "ATARI WINS BY REACHING OTH
ER SIDE";:GOTO 2420
2370 CW=CW+1:POSITION 32,8:? CW;:POSIT
ION 2,22:? "ATARI WINS BY TAKING HUMAN
'S MARKERS";:GOTO 2420
```

Screen wipe for beginning new game.

```
2410 FOR I=1 TO 3:POSITION P(I,0),P(I,
1):? A$;:NEXT I
2412 FOR I=4 TO 6:POSITION P(I,0),P(I,
1):? BL$;:NEXT I
2414 FOR I=7 TO 9:POSITION P(I,0),P(I,
1):? H$;:NEXT I
2416 POSITION 15,10:? "GAME #";Z;:POS
ITION 0,12:? N$;"'S MOVES";:POSITION 2
7,12:? "ATARI'S MOVES";:RETURN
2420 Z=Z+1:FOR Q=1 TO 700:NEXT Q:POS=1
3:GOSUB CLS:POSITION 22,10:? Z;:GOTO 4
60
```

Data needed for possible board positions, random moves, and print positions for markers.

```
2460 DATA 333122211,242536,333212121,1
43536,333221112,262514,233231122,26575
8,332132221,245958,323312221,154736
2465 DATA 323213122,351469,323321212,4
74899,323123212,696899,233132221,24365
8,332231122,261458,233212221,353699
2470 DATA 332212122,141599,332121221,2
42526,233121122,242526,233311122,26359
9,332113221,152499,323122221,369999
2475 DATA 323221122,149999,233212122,3
53699,332212221,141599,323112212,15353
6,323211212,141535,322331222,475899
2480 DATA 223133222,695899,232113222,2
46999,232311222,264799,223331222,47589
9,322133222,695899,232132222,245899
2485 DATA 232231222,265899,223312222,4
73536,322213222,691514,322312222,47159
9,223213222,693599,223111222,359999
2490 DATA 322111222,159999,14,1,18,1,2
2,1,14,4,18,4,22,4,14,7,18,7,22,7
2500 COLOR 32:FOR AZ=POS TO 23:PLOT 0,
AZ:DRAWTO 39,AZ:NEXT AZ:POSITION 0,POS
:RETURN
```

Erase move on last board if all moves in current board lose.

```
2600 D(1)=INT(N(BN)/10000):D=N(BN)-100
00*D(1)
2610 D(2)=INT(D/100)
2620 D(3)=D-100*D(2)
2630 D(MN)=99
2640 IF MN=1 THEN N(BN)=990000+100*D(2
)+D(3)
2650 IF MN=2 THEN N(BN)=10000*D(1)+990
0+D(3)
2660 IF MN=3 THEN N(BN)=10000*D(1)+100
*D(2)+99
2670 GOTO 2420
```

HAVE WE GOT A PROGRAM FOR YOU IN '82

Over 150,000 computer owners and novices attended the 1981 National Computer Shows and Office Equipment Expositions, and more than a quarter of a million are expected to be at the 1982 shows.

Each show features **hundreds** of companies using **thousands** of square feet of display space to showcase and sell **millions** of dollars worth of micro and mini computers, data and word processing equipment, peripherals, accessories, supplies and software.

Under one roof you'll see — and be able to buy — all of the hardware and software made by every major computer manufacturer for business, industry, government, education, home and personal use.

The show includes computers costing as little as $100 to computers selling for $150,000.

Don't miss the coming of the new computers — show up for the show. Admission is $5, and free for children under five years of age.

THE SOUTHWEST COMPUTER SHOW

Dallas
Dallas Market Hall

Thursday-Sunday
April 15-18, 1982
11 AM to 6 PM Daily

DIRECTIONS:
2200 STEMMONS FREEWAY
(AT INDUSTRIAL BLVD)

THE NEW YORK COMPUTER SHOW

Uniondale, Long Island
Nassau Coliseum

Thursday-Sunday
April 22-25, 1982
11 AM to 6 PM Daily

DIRECTIONS: TAKE L.I. EXPWY TO EXIT 38 NO. STATE PKWY TO EXIT 31A MEADOWBROOK PKWY SO. TO EXIT M5 HEMPSTEAD TURNPIKE

THE MID-ATLANTIC COMPUTER SHOW

Washington, DC
DC Armory/Starplex
Across from RFK Stadium

Thursday-Sunday
October 28-31, 1982
11 AM to 6 PM Daily

DIRECTIONS:
2001 E. CAPITOL ST. SE
(E. CAPITOL ST. EXIT OFF I-295 — KENILWORTH FRWY)

THE NATIONAL COMPUTER SHOWS AND OFFICE EQUIPMENT EXPOSITIONS

THE MID-WEST COMPUTER SHOW

Chicago
(Arlington Heights)
Arlington Park Racetrack Exhibition Center

Thursday-Sunday
November 5-7, 1982
11 AM to 6 PM Daily

DIRECTIONS: EUCLID AVE & WILKE RD. TAKE NW TOLLWAY TO RTE 53 EXIT AT EUCLID AVE EAST

THE NORTHEAST COMPUTER SHOW

Boston
Hynes Auditorium/
Prudential Center

Thursday-Sunday
November 11-14, 1982
11 AM to 6 PM Daily

DIRECTIONS: TAKE MASS PIKE TO PRUDENTIAL CENTER EXIT

THE SOUTHEAST COMPUTER SHOW

Atlanta
Atlanta Civic Center

Thursday-Sunday
December 9-12, 1982
11 AM to 6 PM Daily

DIRECTIONS:
395 PIEDMONT AVE NE
(AT RALPH McGILL BLVD)

Ticket Information
Send $5 with the name of the show you plan to attend to National Computer Shows, 824 Boylston Street, Chestnut Hill, Mass. 02167. Tickets can also be purchased at the show.

The National Computer Shows are produced by Northeast Expositions Inc. who also produce Electronica — shows featuring home entertainment equipment and personal electronics — which are held annually in major US cities. NEI also produces the Applefest Shows. For more information about any of these events call us at 617-739-2000 or write to the above address.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         TRS-80 BASIC          $
$          "HEXAPAWN"           $
$ AUTHOR: CARL A. BEVINGTON  $
$    (C) 1982    SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization.

```
140 DEFINTA-Z:CLEAR1000:DIMM$(40),N(40):CLS:PRINTCHR$(23):Y$="BY
 MARTIN GARDNER":K$="H E X A P A W N"
```

Opening graphics cover and text.

```
180 FORX=0TO126STEP2:SET(X,0):SET(X,1):SET(X,19):SET(X,20):SET(X
,46):SET(X,47):NEXTX:FORY=3TO17STEP2:SET(0,Y):SET(1,Y):SET(121,Y
):SET(124,Y):NEXTY:FORY=22TO45STEP2:SET(0,Y):SET(1,Y):SET(121,Y)
:SET(124,Y):NEXTY:PRINT@586,"PROGRAM CREATED BY - ";
190 PRINT@714,"MR. CARL A. BEVINGTON";:FORI=1TO5:PRINT@138,"
                        ";:FORQ=1TO50:NEXTQ:PRINT@144,K$;:PRINT@270
,Y$;:FORQ=1TO500:NEXTQ,I:CLS:FORQ=1TO50:NEXTQ
```

Graphics for instructions.

```
230 PRINT@83,"COMPUTER COMPUTER COMPUTER";:PRINT@340,"PLAYER   P
LAYER   PLAYER";:FORX=36TO91:SET(X,1):SET(X,7):SET(X,13):SET(X,1
9):NEXTX:FORY=1TO19:SET(36,Y):SET(37,Y):SET(54,Y):SET(55,Y):SET(
72,Y):SET(73,Y):SET(92,Y):SET(93,Y):NEXTY
```

Instructions for playing Hexapawn.

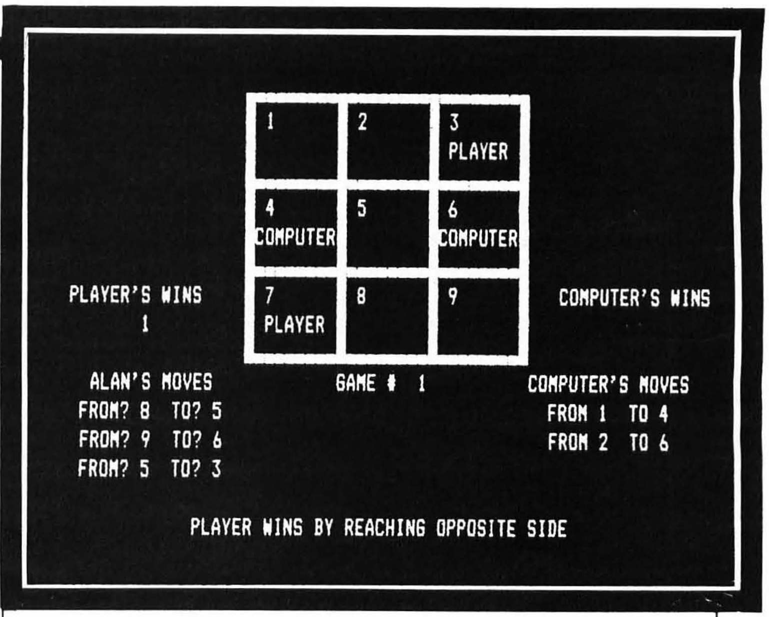
```
270 PRINT@512,"THE GAME OF HEXAPAWN IS PLAYED ON A 3X3 GRID WITH
 THREE MARKERS":PRINT"FOR EACH PLAYER.  THE MARKERS ARE MOVED IN
 THE SAME MANNER AS":PRINT"THE PAWNS IN CHESS:":PRINT"(1) A MARK
ER MAY BE MOVED FORWARD ONE CELL TO AN EMPTY CELL."
280 RANDOM:PRINT"(2) A MARKER MAY BE MOVED FORWARD DIAGONALLY TO
 THE LEFT OR":PRINT"     RIGHT TO CAPTURE AN OPPONENT'S MARKER.
 THE CAPTURED":PRINT"     MARKER IS TAKEN OUT OF PLAY."
```

Read codes for possible board positions, random moves, and print positions for markers.

```
320 Z=1:FORI=1TO37:READA#:M$(I)=STR$(A#):READN(I):NEXTI:FORI=1TO
9:READP(I):NEXTI:PRINT@988,">> PRESS SPACE BAR TO CONTINUE <<";
330 B$=INKEY$:IFB$=" "THEN370ELSE330
```

More instructions for play.

```
370 PRINT@512,"THE GAME IS WON IN THREE POSSIBLE WAYS:
             ";:PRINT:PRINT"(1) BY MOVING A MARKER TO THE OPPON
ENT'S SIDE OF THE GRID.":PRINT"(2) BY CAPTURING ALL OF THE OPPON
ENT'S MARKERS."
380 PRINT"(3) BY PLANNING MOVES SO THAT A POINT IS REACHED IN TH
E GAME":PRINT"     WHERE THE OPPONENT IS UNABLE TO MOVE ANY MARK
ERS.":PRINT
390 B$=INKEY$:IFB$=""THEN390ELSE400
400 PRINT@512,"ALTHOUGH THE COMPUTER IS PROGRAMMED TO MAKE ITS D
IFFERENT MOVES AT RANDOM, THE COMPUTER WILL 'REMEMBER' HOW IT LO
ST A GAME AND  WILL NOT TRY THE SAME RANDOM MOVE AGAIN.
             ":PRINT:PRINT
410 PRINT@960,"
          ";:PRINT@896,"ENTER YOUR FIRST NAME, PLEASE";:INPUTN
$
```



```
          1        2        3
                            PLAYER
          4        5        6
          COMPUTER          COMPUTER
PLAYER'S WINS  7        8        9      COMPUTER'S WINS
     1         PLAYER

 ALAN'S MOVES          GAME #  1      COMPUTER'S MOVES
 FROM? 8   TO? 5                        FROM 1   TO 4
 FROM? 9   TO? 6                        FROM 2   TO 6
 FROM? 5   TO? 3

        PLAYER WINS BY REACHING OPPOSITE SIDE
```

Graphics for playing board.

```
450 CLS:FORQ=1TO50:NEXTQ:FORI=1TO9:PRINT@P(I),I;:NEXTI:FORX=36TO
91:SET(X,1):SET(X,10):SET(X,19):SET(X,28):NEXTX:FORY=1TO28:SET(3
6,Y):SET(37,Y):SET(54,Y):SET(55,Y):SET(72,Y):SET(73,Y):SET(90,Y)
:SET(91,Y):NEXTY:GOSUB2410
460 PRINT@449,"PLAYER'S WINS";:PRINT@497,"COMPUTER'S WINS";:GOSU
B2410:C$="":D$="":E$="":FORJ=1TO7STEP3:FORI=JTOJ+2:C(I)=4-INT((I
+2)/3):NEXTI,J:V=0:W=0:BN=0:MN=0
```

First player move entered here.

```
500 PRINT@706,"                    ";:PRINT@706,"FROM";:INPUTF:F
ORQ=1TO50:NEXTQ:PRINT@715,"TO";:INPUTT
```

Check for valid player move.

```
540 IFF>9ORT>9THEN500
550 IFC(F)<>1THEN500
560 IFC(T)<>2THEN500
570 IFF-T<>3THEN500
```

Set markers on playing board.

```
610 GOSUB2140
620 C(F)=2:C(T)=1
```

Find code for current playing board marker positions.

```
660 FORI=1TO9:C$=STR$(10*VAL(C$)+C(I)):NEXTI
670 FORI=1TO3
680 IFC$=M$(I)THENV=I
690 NEXTI
```

Determine one of three possible random computer moves.

```
730 D(1)=INT(N(V)/10000):D=N(V)-10000*D(1)
740 D(2)=INT(D/100)
750 D(3)=D-100*D(2)
760 G=RND(3)
770 IFD(G)=99THEN760
780 CF=INT(D(G)/10):CT=D(G)-10*CF:W=G
```

Print first computer move.

```
820 PRINT@752,"FROM";CF;:PRINT@760,"TO";CT;
830 FORQ=1TO300:NEXTQ
840 C(CF)=2:C(CT)=3
850 GOSUB2180
```

Second player move entered here.

```
890 PRINT@770,"                         ";:PRINT@770,"FROM";:INPUTF:FOR
Q=1TO50:NEXTQ:PRINT@779,"TO";:INPUTT
```

Check for valid player move.

```
930 IFF>9ORT>9THEN890
940 IFC(F)<>1THEN890
950 IFF-T=3ANDC(T)=2THEN1020
960 IFF-T=2ANDC(T)=3THEN1020
970 IFF-T=4ANDC(T)=3THEN1020
980 GOTO890
```

Set marker on playing board.

```
1020 C(F)=2:C(T)=1:GOSUB2140
1030 IFT<4THEN2220
1040 IFC(1)*C(3)*C(5)=27THEN2310
```

Find code for current playing board marker positions.

```
1080 FORI=1TO9:D$=STR$(10*VAL(D$)+C(I)):NEXTI
1090 BN=V:MN=W:FORI=4TO23
1100 IFD$=M$(I)THENV=I
1110 NEXTI
```

Determine one of three possible random computer moves.

```
1150 D(1)=INT(N(V)/10000):D=N(V)-10000*D(1)
1160 D(2)=INT(D/100)
1170 D(3)=D-100*D(2)
1180 G=RND(3)
1190 IFD(G)=99THEN1180
1200 CF=INT(D(G)/10):CT=D(G)-10*CF:W=G
```

Print second computer move.

```
1240 PRINT@816,"FROM";CF;:PRINT@824,"TO";CT;
1250 FORQ=1TO300:NEXTQ
1260 C(CF)=2:C(CT)=3:W=G
1270 GOSUB2180
```

Check for possible computer win.

```
1310 IFCT>6THEN2360
1320 IFV=12ANDD(G)=36THEN1400
1330 IFV=13ANDD(G)=14THEN1400
1340 IFV=14ANDD(G)=26THEN1400
1350 IFV=15ANDD(G)=24THEN1400
1360 IFV=22ANDD(G)=35THEN1400
1370 IFV=23ANDD(G)=15THEN1400
1380 IFV=18ORV=19THEN1400
1390 GOTO1450
1400 CW=CW+1:PRINT@568,CW;:PRINT@981,"COMPUTER WINS BY BLOCKING"
;:GOTO2420
1410 GOTO460
```

Third player move entered here.

```
1450 PRINT@834,"                         ";:PRINT@834,"FROM";:INPUTF:FO
RQ=1TO50:NEXTQ:PRINT@843,"TO";:INPUTT
```

Check for valid player move.

```
1490 IFF>9ORT>9THEN1450
1500 IFC(F)<>1THEN1450
1510 IFF-T=3ANDC(T)=2THEN1550
1520 IFF-T=2ANDC(T)=3THEN1550
1530 IFF-T=4ANDC(T)=3THEN1550
1540 GOTO1450
1550 C(F)=2:C(T)=1
1560 IFC(2)*C(5)=3ANDC(1)*C(3)*C(4)*C(6)*C(7)*C(8)*C(9)=128THEN2
310
```

Set marker on playing board.

```
1600 GOSUB2140
1610 IFT<4THEN2220
```

Find code for current playing board marker positions.

```
1650 FORI=1TO9:E$=STR$(10*VAL(E$)+C(I)):NEXTI
1660 BN=V:MN=W:FORI=24TO37
1670 IFE$=M$(I)THENV=I
1680 NEXTI
```

Determine one of three possible random computer moves.

```
1720 D(1)=INT(N(V)/10000):D=N(V)-10000*D(1)
1730 D(2)=INT(D/100)
1740 D(3)=D-100*D(2)
1750 G=RND(3)
1760 IFD(G)=99THEN1750
1770 CF=INT(D(G)/10):CT=D(G)-10*CF:W=G
```

Print third computer move.

```
1810 PRINT@880,"FROM";CF;:PRINT@888,"TO";CT;
1820 FORQ=1TO300:NEXTQ
1830 C(CF)=2:C(CT)=3:W=G
1840 GOSUB2180
```

Check for possible computer win.

```
1880 IFCT>6THEN2360
1890 IFV=34ANDD(G)=15THEN2370
1900 IFV=35ANDD(G)=35THEN2370
1910 IFV=30ANDD(G)=24THEN2370
1920 IFV=31ANDD(G)=26THEN2370
1930 IFV=32ANDD(G)=35THEN2370
1940 IFV=33ANDD(G)=15THEN2370
```

Fourth player move entered here.

```
1980 PRINT@898,"                         ";:PRINT@898,"FROM";:INPUTF:FO
RQ=1TO50:NEXTQ:PRINT@907,"TO";:INPUTT
```

Check for valid player move.

```
2020 IFF>9ORT>9THEN1980
2030 IFC(F)<>1THEN1980
2040 IFC(T)<>2THEN1980
2050 IFF-T<>3THEN1980
```

Set marker on playing board.

```
2090 GOSUB2140
```

```
2100 GOTO2220
```

Graphics subroutine for player move.

```
2140 PRINT@P(F)+64,"          ";:PRINT@P(T)+64," PLAYER ";:RETURN
```

Graphics subroutine for computer move.

```
2180 PRINT@P(CF)+64,"          ";:PRINT@P(CT)+64,"COMPUTER";:RETUR
N
```

Player win announcement.

```
2220 PW=PW+1:PRINT@519,PW;:PRINT@973,"PLAYER WINS BY REACHING OP
POSITE SIDE";
```

Reset possible random moves for computer.

```
2260 D(W)=99
2270 IFW=1THENN(V)=990000+100*D(2)+D(3)
2280 IFW=2THENN(V)=10000*D(1)+9900+D(3)
2290 IFW=3THENN(V)=10000*D(1)+100*D(2)+99
2300 IFD(1)+D(2)+D(3)=297THEN2600ELSE2420
2310 GOSUB2140
```

Player/computer win announcements.

```
2350 PW=PW+1:PRINT@519,PW;:PRINT@981,"PLAYER WINS BY BLOCKING";:
GOTO2260
2360 CW=CW+1:PRINT@568,CW;:PRINT@974,"COMPUTER WINS BY REACHING
OTHER SIDE";:GOTO2420
2370 CW=CW+1:PRINT@568,CW;:PRINT@970,"COMPUTER WINS BY TAKING AL
L PLAYER'S MARKERS";:GOTO2420
```

Screen wipe for beginning new game.

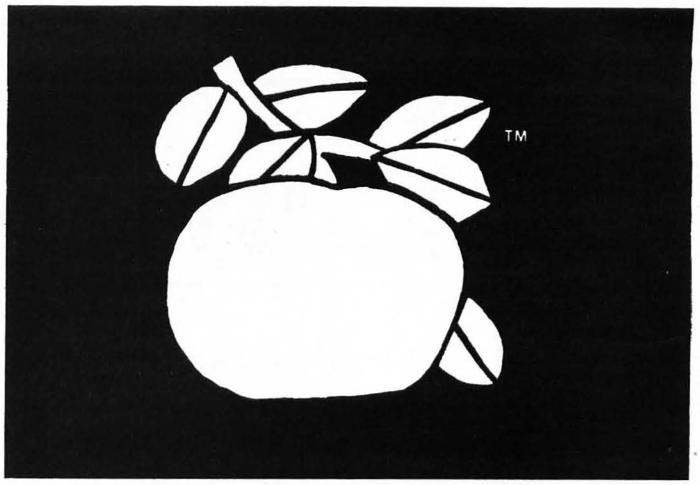
```
2410 FORI=1TO3:PRINT@P(I)+64,"COMPUTER";:PRINT@P(I+3)+64,"
  ";:PRINT@P(I+6)+64," PLAYER ";:NEXTI:PRINT@643,N$;"'S MOVES";:
PRINT@667,"GAME # ";Z;:PRINT@686,"COMPUTER'S MOVES";:RETURN
2420 Z=Z+1:FORQ=1TO1500:NEXTQ:FORK=704TO960STEP64:PRINT@K,"
                                                          ";:NEXT
K:PRINT@674,Z;:GOTO460
```

Data needed for possible board positions, random moves, and print positions for markers.

```
2460 DATA333122211,242536,333212121,143536,333221112,262514,2332
31122,265758,332132221,245958,323312221,154736,323213122,351469,
323321212,474899,323123212,696899,233132221,243658,332231122,261
458,233212221,353699,332212122,141599,332121221
2470 DATA242526,233121122,242526,233311122,263599,332113221,1524
99,323122221,369999,323221122,149999,233212122,353699,332212221,
141599,323112212,153536,323211212,141535,322331222,475899,223133
222,695899,232113222,246999,232311222,264799
2480 DATA223331222,475899,322133222,695899,232132222,245899,2322
31222,265899,223312222,473536,322213222,691514,322312222,471599,
223213222,693599,223111222,359999,322111222,159999,83,92,101,275
,284,293,467,476,485
```

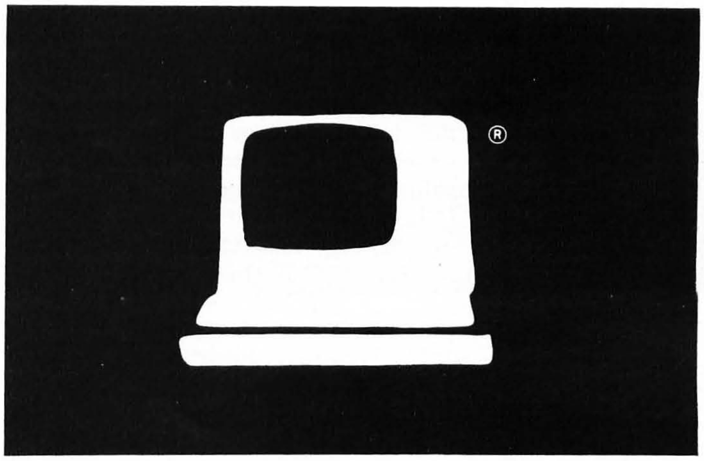
Erase move on last board if all moves in current board lose.

```
2600 D(1)=INT(N(BN)/10000):D=N(BN)-10000*D(1)
2610 D(2)=INT(D/100)
2620 D(3)=D-100*D(2)
2630 D(MN)=99
2640 IFMN=1THENN(BN)=990000+100*D(2)+D(3)
2650 IFMN=2THENN(BN)=10000*D(1)+9900+D(3)
2660 IFMN=3THENN(BN)=10000*D(1)+100*D(2)+99
2670 GOTO2420
```



APPLE™/SIDE page 38

ATARI®/SIDE page 58



TRS-80®/SIDE page 74

K-Bytes

File Matcher

An Applesoft K-Byter by Jon Voskuil, with ATARI® and TRS-80® translations by Alan J. Zett

In the course of writing, revising, and troubleshooting programs, it's fairly common to have more than one version of a program saved on disk at any time. It's quite possible to lose track of what changes have been made, or whether the version on one disk is the same as the version on another disk. This simple utility program will enable you to give your computer the job of comparing the listings. (After all, that's the kind of tedious stuff that computers just love to do.)

The two programs to be compared must first be saved on disk in ASCII format rather than the usual tokenized form. This is accomplished on the TRS-80® by following the SAVE command with a comma and the letter A: SAVE"filename",A. On the ATARI®, instead of using the command SAVE, use the command LIST: LIST"D:filename".

On the Apple, the procedure is a little more complicated (but still well worth it for a program of much substance). You'll need to add a few lines to your program which will cause it to LIST itself to a disk text file. Probably the easiest way to to this is to create a utility text file which you can EXEC from the disk while your program is in memory. Using the MAKE TEXT program on the System Master Diskette, write a text file which contains program lines 32750-32759, below, followed by the command GOTO 32750, and save this as PROGRAM FILER.

```
32750  DIM F$(30):D$ = "": REM CT
     RL-D
32751  CALL  - 936: VTAB 5: PRINT
     "FILE NAME: ";: INPUT F$
32752  PRINT : PRINT : PRINT D$;"
     OPEN ";F$
32753  PRINT D$;"DELETE ";F$
32754  PRINT D$;"OPEN ";F$
32755  PRINT D$;"WRITE ";F$
32756  LIST 0,32749
32757  PRINT D$;"CLOSE ";F$
32758  PRINT : PRINT "--- DONE --
     -"
32759  END
```

These lines will work with both Integer and Applesoft programs having no lines higher than 32749. You may want to create a "File Matcher Utility Disk" which contains the *File Matcher* program along with "PROGRAM FILER". Then, with each of your programs in turn, LOAD them into memory and EXEC PROGRAM FILER.

Once the programs to be compared have thus been saved to disk in ASCII format, you can RUN the *File Matcher* program, supplying it with the names of the two files. It will automatically read each line of the two files, alerting you if there is any difference between them.

A helpful enhancement to the program would be to add a provision for selectively reading ahead one or more lines in either file, in the event of a line mismatch. This would be very useful, for example, if a line has been added to file B which is not in file A. You may want to make a note of that addition and then continue with the comparison by skipping over that added line. A manual "stepping" feature would allow you to do this easily.

APPLE

```
100 D$ =  CHR$ (13) +  CHR$ (4)
110  PRINT D$;"NOMON C,I,O"
120  HOME
130  INPUT "FILE 1: ";F1$
140  VTAB 3: INPUT "FILE 2: ";F2$
150  ONERR  GOTO 360
160 RD$ = D$ + "READ"
170 OP$ = D$ + "OPEN"
180 RTN$ =  CHR$ (13)
190 N = 0
200  VTAB 5: INVERSE : PRINT " CH
     ECKING RECORD -> ";: NORMAL
210  PRINT OP$;F2$
220  PRINT OP$;F1$
230 N = N + 1: PRINT RD$;F1$
240  VTAB 5: HTAB 22: PRINT N;
250 L1$ = "":L2$ = ""
260  GET C$:L1$ = L1$ + C$: IF C$
      < > RTN$ THEN 260
270  VTAB 7: HTAB 1: PRINT  SPC(
     255): VTAB 7: HTAB 1: PRINT
     L1$
280  PRINT RD$;F2$
290  GET C$:L2$ = L2$ + C$: IF C$
      < > RTN$ THEN 290
300  VTAB 15: HTAB 1: PRINT  SPC(
     255): VTAB 15: HTAB 1: PRINT
     L2$
310  IF L1$ = L2$ THEN 230
320  VTAB 22: HTAB 1: PRINT  CHR$
     (7);"RECORDS DO NOT MATCH; C
     ONTINUE? ";: POKE  - 16368,0
330 X =  PEEK ( - 16384): IF X <
     127 THEN 330
340  IF X < > 206 THEN  VTAB 22:
      HTAB 1: PRINT  SPC( 35): GOTO
      230
350  POKE  - 16368,0: PRINT D$;"C
     LOSE": END
360  PRINT D$;"CLOSE": VTAB 22: HTAB
     1: PRINT  CHR$ (7);"DONE"
370  END
```

ATARI®

```
10 GRAPHICS 0:DIM F1$(12),F2$(12)
20 DIM L1$(255),L2$(255),SPC$(255)
30 ? "FILE 1";:INPUT F1$
40 POSITION 2,2:? "FILE 2";:INPUT F2$
50 SPC$(255)=" ":SPC$(1,1)=" "
60 SPC$(2)=SPC$(1):N=0:POSITION 2,4
70 ? " CHECKING RECORD -> ";:TRAP 220
80 OPEN #1,4,0,F1$
90 OPEN #2,4,0,F2$
100 N=N+1:POKE 752,1
110 POSITION 23,4:? N;
120 L1$="":L2$=""
130 INPUT #1,L1$
140 POSITION 2,6:? SPC$:POSITION 2,6:?
 L1$
150 INPUT #2,L2$
160 POSITION 2,14:? SPC$:POSITION 2,14
:? L2$
170 IF L1$=L2$ THEN 100
180 POSITION 2,21:? CHR$(253);"RECORDS
 DO NOT MATCH; CONTINUE?";:POKE 764,25
5
190 X=PEEK(764):IF X=255 THEN 190
200 IF X<>35 THEN POSITION 2,21:? CHR$
(156);:GOTO 100
210 CLOSE #1:CLOSE #2:END
220 POSITION 2,21:? "DONE":GOTO 210
```

TRS-80®

```
10 CLS :CLEAR 1000
20 LINE INPUT"Enter FILESPEC of first program ==> ";F1$ :PRINT
30 LINE INPUT"Enter FILESPEC of second program ==> ";F2$ :PRINT
40 PRINT"***** CHECKING RECORD ==>    *****" :PRINT
50 PRINT STRING$(63,143); :ON ERROR GOTO 200
60 OPEN "I", 1, F1$
70 OPEN "I", 2, F2$
80 N=N+1
90 PRINT @281, N"*****";
100 L1$="" :L2$=""
110 LINE INPUT#1, L1$
120 PRINT @512, STRING$(4,255); :PRINT @512, L1$;
130 LINE INPUT#2, L2$
140 PRINT @768, CHR$(31); :PRINT @768, L2$;
150 IF L1$=L2$ THEN 80
160 PRINT @399, " Records DO NOT MATCH; Continue? ";
170 A$=INKEY$ :IF A$="" THEN 170
180 IF A$<>"N" THEN PRINT @384, STRING$(63,143); :GOTO 80
190 PRINT @960,; :CLOSE :ON ERROR GOTO 0 :END
200 PRINT @1023, " DONE . . ." :RESUME 190
```

Peeker/Poker

by Mike Westerfield

***PEEKER/POKER* is a utility program for the Apple II requiring Applesoft and disk drive. It is included as a bonus program on this month's Apple Disk Version.**

PEEKER/POKER is a command-oriented program written in Applesoft BASIC which allows the user to examine and change sectors on Apple II disks. When the program begins, the user is presented with the first catalog sector of the disk in slot 6, drive 1. All reads and writes are also to slot 6, drive 1. At the bottom of the screen is a "#" character, followed by a cursor. Any of the commands described below can now be entered. All commands must be typed exactly as shown or an error will result. (The program won't stop, it just gives an error.)

An example of the sector display is shown in Figure 1.

READ

This command reads a sector from the disk. It begins by placing the track and sector numbers accessed below the display page. It then asks for the new track and sector numbers. (Track numbers begin at 0 and continue to 34; sector numbers range from 0 to 15.) Invalid track or sector numbers result in an error with a chance to try again. After correct input has been received, the sector is read from the disk and written to the display screen.

WRITE

The syntax for a WRITE command is identical to that for a READ command. After correct input has been received, the sector displayed on the screen is written to the indicated disk sector. Note that the sector does not have to be written to the same track and sector from which it was read.

EDIT

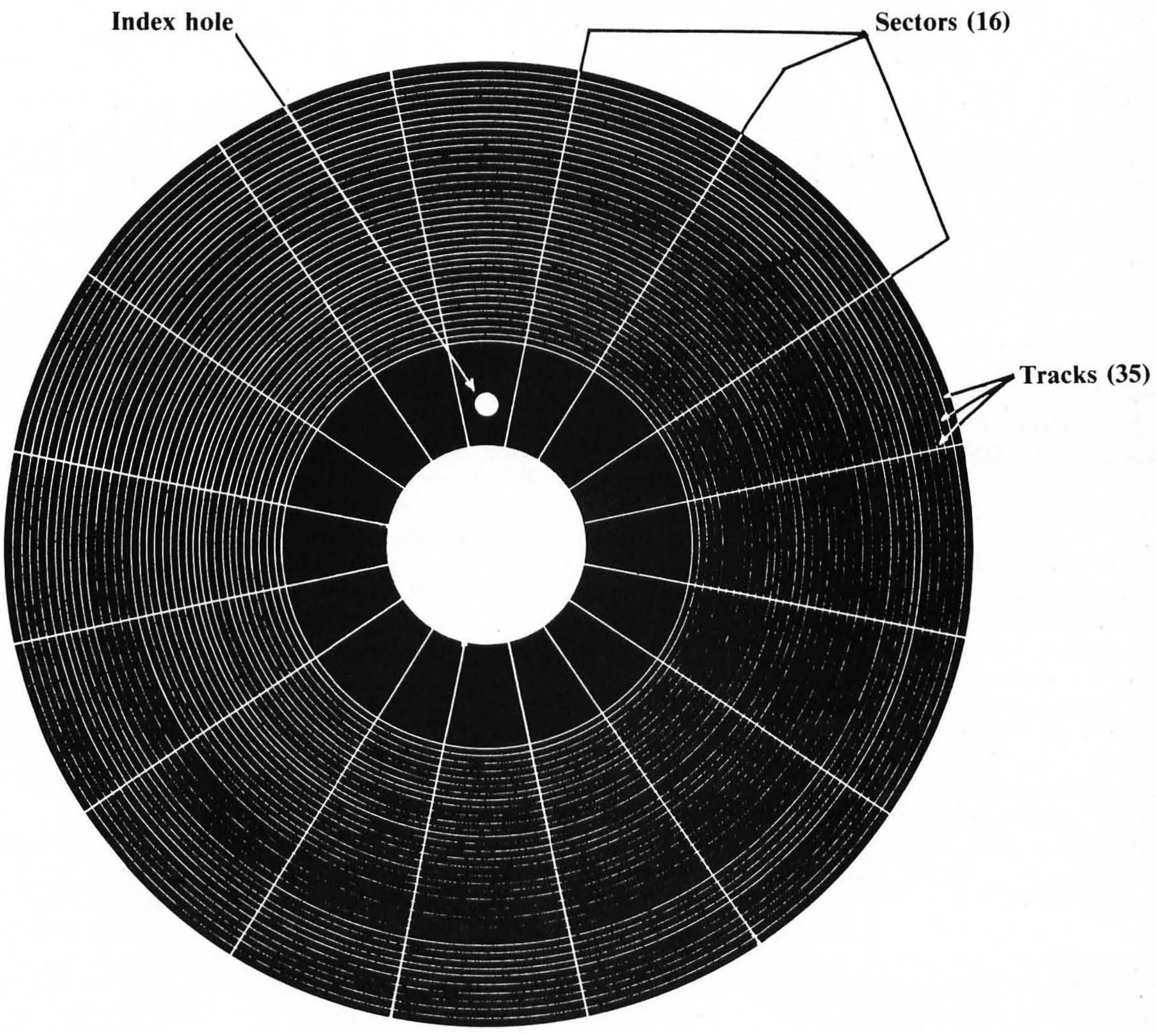
The EDIT command allows you to change the contents of the sector displayed on the screen. Changes made on the screen are not made on the disk until the sector is written to the disk using the WRITE command.

When you enter the edit mode, the cursor is placed over the first nybble of the first byte in the displayed sector. It can be moved using the I, J, K or M keys to move up, left, right or down, respectively. Complete scrolling is provided; if the cursor is moved up from the top

Figure 1

```
00: 00110E00 00000000 00000013 0F82C8C5
10: CCCCCFA0 A0A0A0A0 A0A0A0A0 A0A0A0A0
20: A0A0A0A0 A0A0A0A0 A0A0A0A0 0200140F
30: 84C6C9C4 A0A0A0A0 A0A0A0A0 A0A0A0A0
40: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A014
50: 00150F81 D3C3D2C5 C5CEA0CD C1C3C8C9
60: CEC5A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0
70: A0A02400 160F82C3 C1CCC5CE C4C1D2A0
80: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A0A0A0
90: A0A0A0A0 A0150017 0F82D3C3 D2C5C5CE
A0: A0CDC1C3 C8C9CEC5 A0C6D0A0 A0A0A0A0
B0: A0A0A0A0 A0A0A0A0 2000180F 82C9CED4
C0: C5C7D2C1 D4C9CFCE A0A0A0A0 A0A0A0A0
D0: A0A0A0A0 A0A0A0A0 A0A0A00D 00190F82
E0: D0C5C5CB C5D2AFD0 CFCBC5D2 A0C9C9A0
F0: A0A0A0A0 A0A0A0A0 A0A0A0A0 A0A00B00


#
```



A 16 sector, 35 track floppy disk

line, it shows up in the same column on the bottom line. Moving down from the bottom line places the cursor on the top line. Moving right from the end of a line places you at the start of the next line, and moving left from the start of a line places you at the end of the previous line.

When the cursor is in place, typing a 0 through 9 or A through F changes the nybble that the cursor was on and moves the cursor to the right. The internal buffer where the disk sector is stored is also updated. The edit mode is left by typing a Q. Any key other than those listed (and RESET!) is ignored.

END

This command is used to exit the program. It should be used in lieu of RESET or CTRL-C, since the text window is changed by *PEEKER/POKER* and reset by the END command.

MAKING CHANGES

As listed, the program is set up for DOS 3.3. If DOS 3.2 is to be used, change the sixth data element in line 240 from a 15 to a 12, and change the 15 in line 3050 to a 12.

Variables

A$: Used for input statments.
DT: Address of the 256-byte data table where the disk sector is stored.
HX$: Array containing the hexadecimal characters.
I, J, K, L: General use and indexing variables.
IO: Address of the read/write command flag contained in the IOB. 1 indicates a read, 2 indicates a write.
SK: Address of sector number to read or write.
TK: Address of track number to read or write.

Magical Shape Machine

by Tom Keith

***The Magical Shape Machine* is a graphics utility for an Apple with Applesoft in ROM and 48K RAM. (It may be modified for 32K systems.)**

Have you ever marvelled at all those fascinating shapes and graphics images that float around your Apple's video screen?

And have you ever tried to create your own shapes, by following the instructions in the various Apple manuals? And have you given up, because you were about to go cross-eyed writing down scores of little arrows? Don't you dread having to convert those arrows into hexadecimal code?

Well, salvation is near — it's *The Magical Shape Machine*! Now, just decide what you want your shapes to look like (with a few easy keystrokes), and your shapes can spin, glide, shrink and expand before your very eyes. Even better, you can incorporate your shapes into your own programs, quickly and painlessly. You needn't ever look at one of those pesky arrows again!

If you've got all the necessary equipment and don't need modifications, just run the program. The menu will tell you what you can do.

Drawing Shapes

At first, your choice is between beginning a new shape and loading an old shape table. Since you've never run the program before, you probably don't have any shape tables saved on your disk. So choose B (begin new shape). First, the computer will ask you how many points you want to cover with each keystroke. The higher the number you choose, the easier your drawing will be — but the shapes will ultimately be smaller, and not as finely detailed. To get the hang of it at first, try 4 or 8.

This puts you on the Hi-Res screen. Your choices are listed at the bottom of the screen; they're based on on the I-J-K-M pattern that you're probably already familiar with. If not (or as a reminder), "I" will draw a dot or a line (depending upon which scale you chose) upward, "J" will draw to the left, "K" will draw to the right, and "M" will draw down. If you'd like to make a move without drawing, just hold the CTRL key down as you press I, J, K, or M.

The flashing dot will show you where you are, and your position on the screen is also indicated by the X-Y coordinates at the bottom of the screen.

It's a good idea not to draw over a line already drawn in the course of creating your shape. The reason for this is if you use the XDRAW statement (rather than DRAW) in a program to draw the shape, the line will be erased when it is drawn over itself. If you have to go over a line already drawn, hold down the CTRL key while doing so and everything will come out fine. (Doing this may make the line look dotted, or change its intensity. Don't worry — it will look fine when redrawn.)

Another restriction results from the way in which the Apple stores shape tables in memory. Moving the flashing dot up several times in a row without plotting will give unexpected results. Although the shape will look normal when you're drawing it, it may suffer from some vertical compression when viewed later.

The maximum number of CTRL-I moves in succession is two. This may mean resorting to a zig-zag path when moving upward to start a new line, to make the shape come out properly. Alternatively, you should try to plan your shapes so that they avoid the problem in the first place.

Make a mistake? No problem, just press @. This will erase the effect of your last keystroke. (Hold REPT and press @ if your problem is back a few steps).

Once you've got a shape you're happy with, press CTRL-S, to save the shape to your shape tables. Or, you can go back to the menu at any time by pressing ESC. (Note that almost all the program input uses the "GET" statement, so you won't need to hit RETURN after hitting your response.) Also, going back to the menu and then choosing B again is a quick way to erase the whole shape and start over.

Now that you've got something in your shape table, you'll notice that the menu will allow you to do a lot more than it did the first time. You can begin another shape and add it to the table, or you can view the shapes, print out the shape table (to a printer, if you've got one), delete a shape from the table, or manipulate the shapes.

Shape Manipulation

If you choose "manipulate,"

Also Vector Shapes, Hplot-Shapes, Text File Shapes; ANIMATION of many types, including Logical Shift, XDRAW, Page-Flipping; Instant Graphics and Shape-Drawing Programs; Machine Language Animation for all types of shapes; Font Programs; Machine Language Sounds; Hi-res Scrolling; Color Animation; Music (Write:Record:Play); YTABLE; Assembling; Utilities; *HI-RES COLOR PALETTE, PAINTBRUSH & COLOR-FILL PROGRAMS*

BY DON FUDGE

"IF YOU CAN'T BUDGE IT, FUDGE IT"

4 Disks

THE SECRET IS OUT...... **For your Apple*.**

$125, 4 disks, over 200 pages of documentation. Requires Applesoft* in ROM, 48K, DOS 3.2 or 3.3, Disk.

$10.00 HI-RES SECRETS Demo Disk.

ASK YOUR DEALER OR ORDER DIRECT FROM:

AVANT-GARDE CREATIONS, P.O. Box, 30160, Eugene, OR 97403 (503) 345-3043

VISA Visa/Mastercard accepted. MasterCard

*Apple is a registered trademark of Apple Computer Inc.

you'll find yourself transported to another menu, where all sorts of magical things can happen. First, you'll be asked which shape you want to manipulate. Just enter the shape number and hit RETURN. (This is one of the few places in the program where you'll need to use the RETURN key.) If you're not sure which shape has which number, just hit RETURN (to get back to the main menu) and then "V" to scan through the shapes quickly.

When the program knows which shape you want to work with, it will present a list of options. S will allow you to expand or shrink the shape, M will allow you to move it around the screen (use the REPT key for faster motion), and R will let you rotate it. Note that the larger the scale, the finer the rotations will be.

To draw another shape on the screen (without erasing what's already there), press 0. The computer will ask you again which shape you want to manipulate. If you choose a shape that's already on the screen, it won't be redrawn. But you will be able to rotate, scale, erase, etc., using the shape as it's already drawn.

E will let you erase the shape. (Pressing E again will redraw it.) And if the screen is getting just too cluttered, press C to clear off everything. Or if you're getting tired of manipulating the shapes, press # to get back to the main menu.

Other Features

From the main menu, you can also delete any shape in the table, by stepping through your shapes until you find the one you want to delete. The remaining shapes will be moved down one number, and you'll be able to draw more shapes at the end of the table later.

If you choose the print option, you can look at the elements of the table (in decimal), and print them to your printer. The table is printed out in the form of a DATA line that you can read and POKE into your own programs. If you don't have a disk drive, you can use this to copy short shape tables right off the screen. Just hit ESC to end the program, type "NEW", then move the cursor to the beginning of the DATA statement. Hold down the right arrow and REPT keys to copy the line; then write the rest of your program, incorporating a READ routine to POKE the table into memory. If the shape table is more than 245 bytes, you'll have to divide it into two or more DATA statements.

Best of all, from the menu you can save your shape table onto a disk, either to use it in your own program, or to add more shapes (or delete any you don't like) later. It will be saved under the name you give it, with the prefix "SHAPE:". To load it back into *The Magical Shape Machine* later, you don't need to type "SHAPE:"; but to BLOAD the table into your own programs, you will need the prefix. In your own programs, you'll also have to POKE 232,0 and POKE 233,64 to tell the Apple where the table starts. You'll also have to avoid using the second page of Hi-Res graphics, unless you BLOAD the table elsewhere, and POKE the appropriate values into 232 and 233.

Program Modifications

If you have only 32K, LOAD the program, type in the following line, and SAVE the program:

```
1750 I = PEEK (27232) + 256 *
PEEK (27233) + 1 :I1 = I: GOTO
100
```

If you don't want your shape table stored above the first Hi-Res page, just change the value of T in line 14 to the decimal equivalent of the location where you want the table to begin. Note that if that location is above the top of the second Hi-Res page, you'll have to raise LOMEM, too (line 1).

If "PR#1" won't work to call your printer, you'll have to change lines 1030 and 1060.

If you plan on more than 20 shapes in your table, change N in line 12.

Other than those simple modifications, be careful when changing any code. The program just barely squeezes below Hi-Res page 1 as is. Any changes may cause you to lose part of the delete routine.

If you don't have a disk drive, you won't be able to save and retrieve shape tables. But you can still store the tables on tape — see page 97 of the Applesoft manual for details. (The starting address you'll need is 4000 in hexadecimal; the length of the table is given in decimal if you choose the print option from the menu.)

Variables

A, C, C1, C2, C3, D, J, L, X1, Y1: Scratch variables.
A$, A1$: Keyboard input.
A1: Shape number being manipulated (lines 2500-2960); otherwise, a scratch variable.
AA$: Name of shape table being BSAVEd or BLOADed.
B: Shape number being drawn or deleted; otherwise, a scratch variable.
D(A1): Array to check whether shape is DRAWn on screen.
D$: Alerts computer for a DOS command.
I: End of shape table.
I1: End of last shape/previous end of shape table.
INCR: Number of points per keystroke.
LODED: Flag to check whether last shape on Hi-Res screen has been POKEd into memory with the correct shape table index (1 = it has).
N: Maximum number of shapes allowed.
R(A1): Current rotation for shape being manipulated.
SC(A1): Current scale for shape being manipulated.
SHAPE: Total number of shapes currently in shape table.
SS: Flag to determine whether current shape table has been saved (1 = saved).
T: Memory location for start of shape table.
X, Y: Current horizontal and vertical coordinates being drawn.
X(A1), Y(A1): Horizontal and vertical coordinates of shape being manipulated.

Additionally, two functions are used:

FN P(N): PEEKs into the Nth memory location in the shape table.
FN Q(N): Finds the starting address for shape N in the shape table.

APPLE™

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$  'MAGICAL SHAPE MACHINE'     $
$ AUTHOR: THOMAS W. KEITH JR $
$    (C) 1982    SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Set LOMEM to protect Hi-Res graphics screen, and initialize variables.

```
1  LOMEM: 24576:I1 = 2: GOSUB 10:
     GOTO 100
10 SHAPE = 0:D$ =  CHR$ (13) +  CHR$
     (4):LODED = 1: DEF  FN P(N) =
     PEEK (T + N): DEF  FN Q(N) =
     PEEK (T + 2 * N) + 256 *  PEEK
    (T + 2 * N + 1)
12 N = 20: DIM D(N),SC(N),R(N),X(
    N),Y(N)
14 T = 16384: POKE 233, INT (T /
    256): POKE 232,T - 256 *  PEEK
    (233): RETURN
```

Handle disk errors.

```
20  PRINT : PRINT "FILE NOT FOUND
     ": POKE 216,0: GOSUB 40: GOTO
     100
```

Message to warn user that the shape is off the screen.

```
30  HOME : HTAB 13: FLASH : PRINT
     "EDGE OF SCREEN": NORMAL : GOSUB
     40: GOTO 500
```

Delay loop.

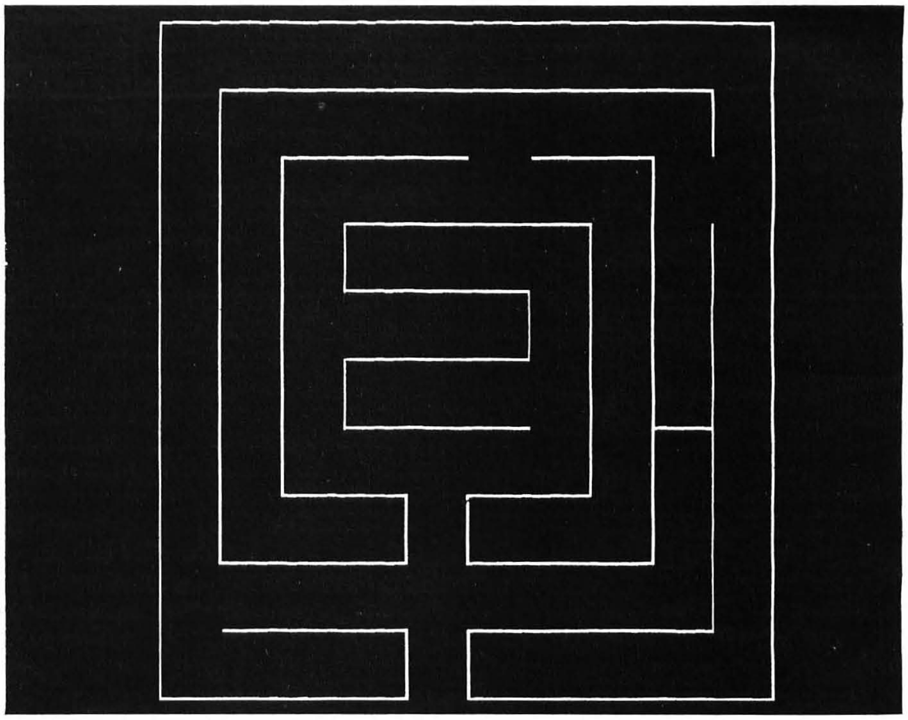
```
40  FOR J = 0 TO 2000: NEXT J: RETURN
```

Initialize the arrays for drawing shapes from the shape table in memory.

```
50 X(A1) = 140:Y(A1) = 80:D(A1) =
     0:SC(A1) = 1:R(A1) = 0: RETURN
```

Inform user of the current status of various shape parameters when they're being manipulated.

```
60  PRINT "SCALE=";SC(A1); SPC( 5
     );"ROT=";R(A1); SPC( 5);"X="
     ;X(A1); TAB( 34);"Y=";Y(A1):
     RETURN
```



Set the correct rotation and scale, and draw or erase the shape being manipulated.

```
70  ROT= R(A1): SCALE= SC(A1): IF
     D(A1) THEN 76
73 D(A1) =  NOT D(A1): HCOLOR= 3 *
     D(A1): DRAW A1 AT X(A1),Y(A1
     )
76  RETURN
```

Warn user that he's moving his shape off the screen.

```
80  HOME : HTAB 13: FLASH : PRINT
     "EDGE OF SCREEN": NORMAL : GOSUB
     40: RETURN
```

Warn user if a menu choice will erase something already drawn.

```
90  IF  NOT LODED THEN  HTAB 4: PRINT
     "(ERASES SHAPE ON SCREEN)"
95  RETURN
```

Main program menu.

```
100  TEXT : HOME : HTAB 7: PRINT
     "THE MAGICAL SHAPE MACHINE!"
     : PRINT : PRINT : IF  NOT LO
     DED THEN  PRINT "(R) RETURN
     TO LAST SHAPE"
110  IF SHAPE < N THEN  PRINT "(B
     ) BEGIN NEW SHAPE": GOSUB 90
120  PRINT "(L) LOAD OLD SHAPE TA
     BLE": GOSUB 90: IF SHAPE = 0
      THEN 140
130  PRINT "(M) MANIPULATE SHAPES
     ": GOSUB 90: PRINT "(V) VIEW
      SHAPES": GOSUB 90: PRINT "(
     D) DELETE A SHAPE": GOSUB 90
     : PRINT "(P) PRINT SHAPE TAB
     LE": PRINT "(S) SAVE SHAPE T
     ABLE"
140  PRINT "(Q) QUIT"
150  PRINT : PRINT "WHICH? ";
200  GET A$: IF A$ = "R" AND  NOT
     LODED THEN 1100
210  IF A$ = "B" AND SHAPE < N THEN
     400
220  IF A$ = "L" THEN 1700
230  IF A$ = "Q" THEN 900
240  IF SHAPE = 0 THEN 200
250  IF A$ = "P" THEN 1000
260  IF A$ = "S" THEN 1500
270  IF A$ = "V" THEN LODED = 1: GOTO
     2000
280  IF A$ = "D" THEN LODED = 1: GOTO
     3000
290  IF A$ = "M" THEN LODED = 1: GOTO
     2500
300  GOTO 200
```

POKE each line or point into the shape table.

```
350  IF C = 1 THEN C = 2:B = 3: GOSUB
      360:B = 1: GOSUB 365: RETURN
355  IF C = 2 THEN C = 0:I = I +
      1
360  IF C = 2 THEN  POKE T + I, FN
      P(I) + 64 * B:I = I + 1:C =
      0: GOTO 375
365  IF C = 0 THEN  POKE T + I,B:
      C = 1: GOTO 375
370  POKE T + I, FN P(I) + 8 * B:
      C = 2
375  RETURN
```

Main drawing routine.

```
400  VTAB 21: HTAB 1: PRINT "DRAW
      1, 2, 4, OR 8 POINTS PER KE
      YSTROKE?": PRINT "CHOOSE 1,2
      ,4,OR 8 : ";
410  GET A$:INCR =  VAL (A$): IF
      INCR <  > 1 AND INCR <  > 2 AND
      INCR <  > 4 AND INCR <  > 8 THEN
      410
420  HGR : POKE 34,20:X = 140:Y =
      80:LODED = 0:I = I1:C = 0:L =
      0
500  HOME : INVERSE : HTAB 12: PRINT
      "X=";X;: HTAB (25): PRINT "Y
      =";Y: NORMAL : PRINT "I=UP
         J=LEFT      K=RIGHT      M=
      DOWN": PRINT "ALSO PRESS CTR
      L TO MOVE WITHOUT DRAWING": PRIN
      "ESC=MENU   CTRL S=SAVE SHAP
      E     @=ERASE";
505  POKE  - 16368,0
510  HCOLOR= 3 * L: HPLOT X,Y:A =
       PEEK ( - 16384): IF A < 128
       THEN L =  NOT L: GOTO 510
520  POKE  - 16368,0: HCOLOR= 0: HPLOT
      X,Y: HCOLOR= 3: IF A > 136 AND
      A < 142 THEN A = A - 136: GOTO
      590
530  IF A > 200 AND A < 206 THEN
      A = A - 200: GOTO 580
540  IF A = 155 THEN 100
550  IF A = 192 THEN  GOSUB 800: GOTO
      500
560  IF A = 147 THEN 1200
570  GOTO 510
580  ON A GOTO 600,620,640,510,66
      0
590  ON A GOTO 680,700,720,510,74
      0
600  IF Y - INCR < 0 THEN 30
610 Y1 = Y:Y = Y - INCR: HPLOT X,
      Y1 TO X,Y:B = 4: GOSUB 355: GOTO
      500
620  IF X - INCR < 0 THEN 30
630 X1 = X:X = X - INCR: HPLOT X1
      ,Y TO X,Y:B = 7: GOSUB 355: GOTO
      500
640  IF X + INCR > 279 THEN 30
650 X1 = X:X = X + INCR: HPLOT X1
      ,Y TO X,Y:B = 5: GOSUB 355: GOTO
      500
660  IF Y + INCR > 159 THEN 30
670 Y1 = Y:Y = Y + INCR: HPLOT X,
      Y1 TO X,Y:B = 6: GOSUB 355: GOTO
      500
680  IF Y - INCR < 0 THEN 30
690 Y = Y - INCR:B = 0: GOSUB 350
      : GOTO 500
700  IF X - INCR < 0 THEN 30
710 X = X - INCR:B = 3: GOSUB 360
      : GOTO 500
720  IF X + INCR > 279 THEN 30
730 X = X + INCR:B = 1: GOSUB 360
      : GOTO 500
740  IF Y + INCR > 159 THEN 30
750 Y = Y + INCR:B = 2: GOSUB 360
      : GOTO 500
```

Erase what has been drawn, point by point, in case of an error.

```
800  IF I = I1 AND C = 0 THEN  HOME
      : PRINT : PRINT "NOTHING TO
      ERASE!": GOSUB 40: GOTO 880
805 X1 = X:Y1 = Y
810  ON C GOTO 840,845
815 I = I - 1:C = 2: IF  FN P(I) <
      64 THEN 845
820  IF  FN P(I) < 128 THEN X = X
      - INCR: GOTO 835
825  IF  FN P(I) - 128 < 64 THEN
      Y = Y - INCR: GOTO 835
830 X = X + INCR
835  POKE T + I, FN P(I) - 64 *  INT
      ( FN P(I) / 64): GOTO 875
840 D =  FN P(I): GOSUB 850: POKE
      T + I,0:C = 0: GOTO 880
845 D =  INT ( FN P(I) / 8): GOSUB
      850: POKE T + I, FN P(I) - 8
      * D:C = 1: GOTO 880
850  IF D = 0 OR D = 4 THEN Y = Y
      + INCR: GOTO 870
855  IF D = 1 OR D = 5 THEN X = X
      - INCR: GOTO 870
860  IF D = 2 OR D = 6 THEN Y = Y
      - INCR: GOTO 870
865 X = X + INCR: GOTO 870
870  IF D > 3 THEN  HCOLOR= 0: HPLOT
      X,Y TO X1,Y1
875  PRINT  CHR$ (7)
880  RETURN
```

The quit routine. Before quitting, a check is made to see if the current shape table has been saved.

```
900  IF SS THEN  PRINT : PRINT "S
      AVE CURRENT SHAPE TABLE? ";:
       GET A$: IF A$ = "Y" THEN 15
      00
910  HOME : END
```

Routine to print the shape table data to the screen or a printer.

```
1000  HOME : PRINT "PRINT OUT SHA
      PE TABLE TO:": PRINT "(S) SC
      REEN": PRINT "(P) PRINTER": PRINT
      : PRINT "WHICH? ";
1010  GET A$: IF A$ = "S" THEN  HOME
      : GOTO 1040
1020  IF A$ <  > "P" THEN 1010
1030  PRINT : PRINT D$"PR#1"
1040  PRINT "SHAPE TABLE IS ";I +
      1;" BYTES LONG.": PRINT "SHA
      PE TABLE DATA (DECIMAL) :": PRINT
      : PRINT "10 DATA ";: FOR J =
      0 TO I: PRINT  FN P(J);",";
1050  IF  PEEK (37) = 21 AND A$ =
      "S" THEN  PRINT : PRINT "HIT
       ANY KEY TO CONTINUE";: GET
      A1$: HOME
1060  NEXT : IF A$ = "P" THEN  PRINT
      D$"PR#0"
1070  PRINT : PRINT "HIT <ESC> TO
       END; ANY OTHER KEY TO
      RETURN TO MENU.";
1080  GET A$: IF A$ <  >  CHR$ (2
      7) THEN 100
1090  END
```

Return to a partially completed shape table (after using ESC to get to the main menu).

```
1100  POKE 34,20: POKE  - 16304,0
      : GOTO 500
```

Compute the shape table index and POKE it into memory.

```
1200 SHAPE = SHAPE + 1
1210 B = 2 * SHAPE: FOR J = I TO
     B STEP  - 1: POKE T + J + 2,
      FN P(J): NEXT :I = I + 2: POKE
     T,SHAPE: POKE T + 1,0: IF SH
     APE = 1 THEN 1250
1220  FOR J = 1 TO SHAPE - 1:X1 =
      FN Q(J) + 2: IF X1 < 256 THEN
      POKE T + 2 * J,X1: POKE T +
     2 * J + 1,0: GOTO 1240
1230  POKE T + 2 * J + 1, INT (X1
      / 256): POKE T + 2 * J,X1 -
     256 *  FN P(2 * J + 1)
1240  NEXT
1250 X1 = I1 + 2: IF X1 < 256 THEN
      POKE T + 2 * SHAPE,X1: POKE
     T + 2 * SHAPE + 1,0: GOTO 12
     70
1260  POKE T + 2 * SHAPE + 1, INT
     (X1 / 256): POKE T + 2 * SHA
     PE,X1 - 256 *  PEEK (T + 2 *
     SHAPE + 1)
1270 I = I + 1: POKE T + I,0
1280 I1 = I + 1:SS = 1:LODED = 1:
      GOTO 100
```

Save the current shape table to disk.

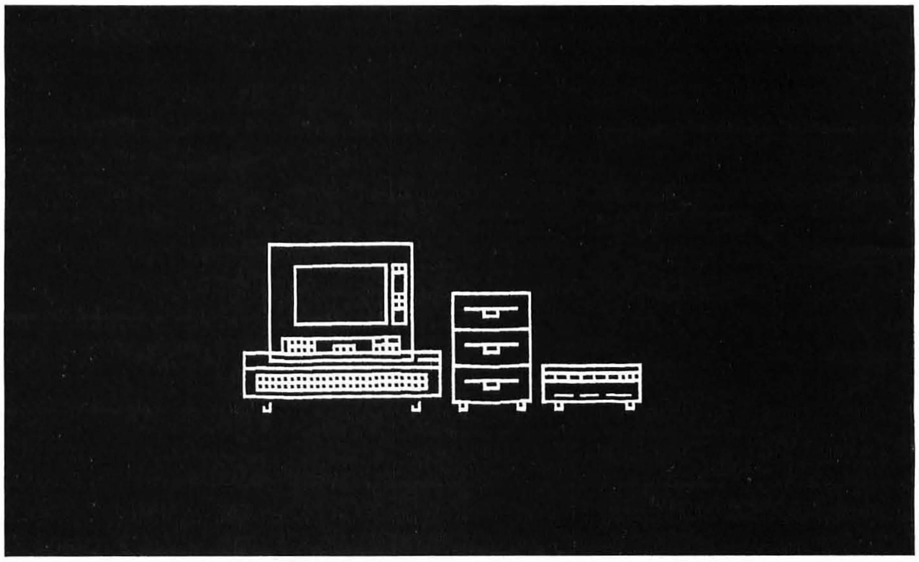
```
1500  PRINT : INPUT "NAME OF SHAP
     E TABLE? ";AA$:AA$ = "SHAPE:
     " + AA$
1510  PRINT D$;"BSAVE";AA$;",A";T
     ;",L";I + 1
1520 SS = 0: GOTO 100
```

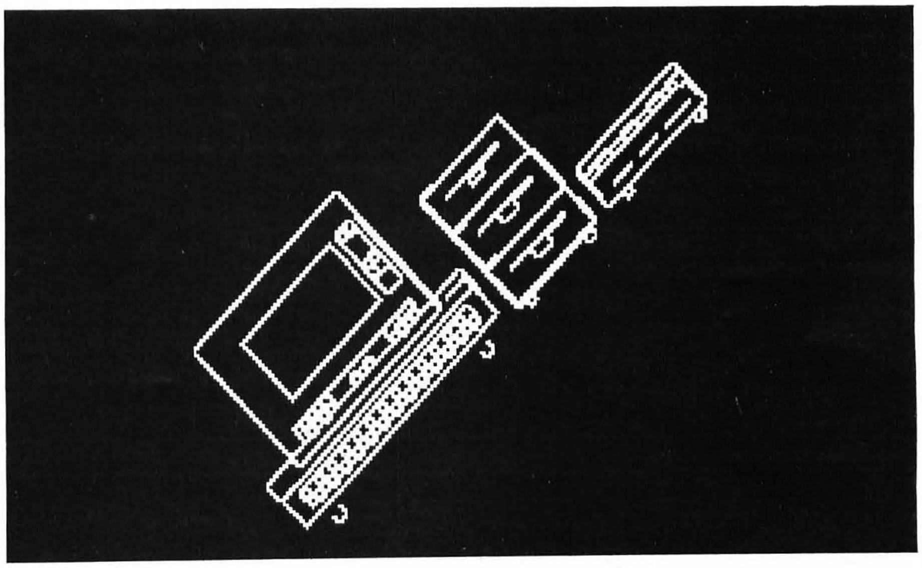
Load a previously saved shape table.

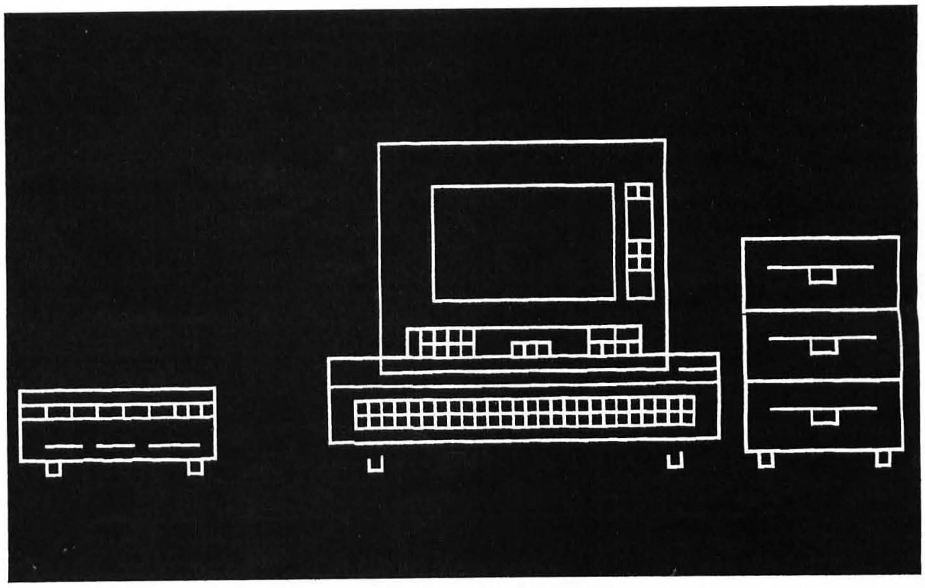
```
1700  CLEAR : GOSUB 10: PRINT : PRINT
     : INPUT "NAME OF SHAPE TABLE
     ? ";AA$:AA$ = "SHAPE:" + AA$

1710  ONERR  GOTO 20
1720  PRINT D$;"BLOAD";AA$
1730  POKE 216,0
1740 SS = 0:SHAPE =  PEEK (T):LOD
     ED = 1
1750 I =  PEEK (43616) + 256 *  PEEK
     (43617) + 1:I1 = I: GOTO 100
```

Display the shapes in the current shape table, in order.







```
2000  HGR : HOME : VTAB 23: HTAB
     4: PRINT "HIT ANY KEY TO SKI
     P THIS SHAPE"
2010  FOR A1 = 1 TO SHAPE: GOSUB
     50: GOSUB 70: VTAB 21: HTAB
     16: INVERSE : PRINT "SHAPE "
     ;A1: NORMAL
2020  FOR J = 0 TO 500: IF  PEEK
     ( - 16384) > 127 THEN J = 50
     0: POKE  - 16368,0
2030  NEXT J: GOSUB 73: NEXT
2040  GOTO 100
```

Manipulate the shapes on the screen.

```
2500  FOR A1 = 1 TO SHAPE: GOSUB
     50: NEXT : HGR : POKE 34,20
2510  HOME : PRINT "THERE ARE ";S
     HAPE;" SHAPES IN THE TABLE."
     : PRINT "MANIPULATE WHICH ON
     E?": PRINT "(HIT RETURN TO R
     ETURN TO MENU)"
2530  INPUT "WHICH? ";A$: IF A$ =
     "" THEN 100
2550 A1 =  VAL (A$): IF A1 < 1 OR
     A1 > SHAPE THEN 2510
2560  GOSUB 70
2600  HOME : INVERSE : PRINT "SHA
     PE ";A1;: HTAB 18: NORMAL : PRINT
     "#=RETURN TO MENU": PRINT "R
     =ROTATE SHAPE   D=DRAW ANOTH
     ER SHAPE": PRINT "E=ERASE SH
     APE    S=CHANGE SCALE": PRINT
     "M=MOVE SHAPE     C=CLEAR SC
     REEN";: HTAB 40
2610  GET A$: IF A$ = "#" THEN 10
     0
2620  IF A$ = "C" THEN 2500
2630  IF A$ = "D" THEN 2510
2640  IF A$ = "E" THEN  GOSUB 73:
      GOTO 2600
2650  IF A$ = "S" THEN 2700
2660  IF A$ = "R" THEN 2800
2670  IF A$ = "M" THEN 2900
2680  GOTO 2610
```

Change the scale.

```
2700  HOME : GOSUB 60: PRINT "RIG
     HT ARROW TO DECREASE SCALE
                LEFT ARROW TO INCRE
     ASE SCALE             ESC FOR
      ANYTHING ELSE";
2710  GET A$: IF A$ =  CHR$ (27) THEN
      2600
2720  IF A$ =  CHR$ (8) THEN 2750

2730  IF A$ =  CHR$ (21) THEN 277
     0
2740  GOTO 2710
2750 SC(A1) = SC(A1) + 1: IF SC(A
     1) > 255 THEN SC(A1) = 255
2760  GOTO 2780
2770 SC(A1) = SC(A1) - 1: IF SC(A
     1) < 1 THEN SC(A1) = 1
2780  GOSUB 73: GOSUB 70: GOTO 27
     00
```

Rotate the shape.

```
2800  HOME : GOSUB 60:L =  INT (1
     6 / SC(A1)): IF SC(A1) > 16 THEN
     L = 1
2810  PRINT "RIGHT ARROW TO ROTAT
     E RIGHT                LEFT ARR
     OW TO ROTATE LEFT
          ESC FOR ANYTHING ELSE";
2820  GET A$: IF A$ =  CHR$ (27) THEN
      2600
2830  IF A$ =  CHR$ (21) THEN R(A
     1) = R(A1) + L: GOTO 2870
2840  IF A$ =  CHR$ (8) THEN R(A1
     ) = R(A1) - L: GOTO 2860
2850  GOTO 2820
2860  IF R(A1) < 0 THEN R(A1) = 6
     4 - L
2870  IF R(A1) >  = 64 THEN R(A1)
      = 0
2880  GOSUB 73: GOSUB 70: GOTO 28
     00
```

Move the shape on the screen.

```
2900  HOME : GOSUB 60: PRINT "I=U
     P; J=LEFT; K=RIGHT; M=DOWN
                ESC=ANYTHING ELSE";
2910  GET A$: IF A$ =  CHR$ (27) THEN
      2600
2920  IF A$ < "I" OR A$ > "M" OR
     A$ = "L" THEN 2910
2930  GOSUB 73:A =  ASC (A$) - 72
     : ON A GOTO 2960,2940,2950,2
     910,2970
2940 X(A1) = X(A1) - 1: IF X(A1) <
      0 THEN X(A1) = 0: GOSUB 80
2945  GOTO 2980
2950 X(A1) = X(A1) + 1: IF X(A1) >
      279 THEN X(A1) = 279: GOSUB
      80
2955  GOTO 2980
2960 Y(A1) = Y(A1) - 1: IF Y(A1) <
      0 THEN Y(A1) = 0: GOSUB 80
2965  GOTO 2980
2970 Y(A1) = Y(A1) + 1: IF Y(A1) >
      191 THEN Y(A1) = 191: GOSUB
      80
2980  GOSUB 70: GOTO 2900
```

Delete a shape from the current shape table.

```
3000  SCALE= 1: ROT= 1:SS = 1:K =
      SHAPE
3010  FOR A1 = 1 TO K: HGR : HOME
      : GOSUB 50: GOSUB 70: VTAB 2
      1: HTAB 16: INVERSE : PRINT
      "SHAPE ";A1: NORMAL : PRINT
      : PRINT "DELETE THIS SHAPE F
      ROM TABLE? (Y/N) ";
3020  GET A$: IF A$ = "Y" THEN 30
      50
3030  IF A$ <  > "N" THEN 3020
3040  NEXT A1: GOTO 100
3050  HOME : VTAB 21: PRINT "HIT
      <D> TO CONFIRM DELETION": PRINT
      "ESC TO GO TO NEXT SHAPE";
3060  GET A$: IF A$ =  CHR$ (27) THEN
      3040
3070  IF A$ <  >  CHR$ (68) THEN
      3060
3080 C1 =  FN Q(A1): IF A1 = SHAP
      E THEN C2 = I + 1: GOTO 3100

3090 C2 =  FN Q(A1 + 1)
3100 C3 = C2 - C1: FOR J = 1 TO S
      HAPE: IF J = A1 THEN 3130
3110 D =  FN Q(J) - 2:L = T + 2 *
      J: IF J > A1 THEN L = L - 2:
      D = D - C3
3120  POKE L + 1, INT (D / 256): POKE
      L,D - 256 *  PEEK (L + 1)
3130  NEXT : FOR J = T +  FN Q(1)
      + 2 TO T + C1 - 1: POKE J -
      2, PEEK (J): NEXT : FOR J =
      T + C2 TO T + I: POKE J - C3
      - 2, PEEK (J): NEXT
3140 I = I - C3 - 2:SHAPE = SHAPE
      - 1: POKE T,SHAPE:A1 = K: GOTO
      3040
```

Gravity-Float Trace

by Robert A. Pritchett

***Gravity-Float Trace* is a color graphics game for an Apple with Applesoft, game paddles, and 16K RAM.**

Are you tired of defending the Earth from Klingons and other evil aliens? Are you tired of running for your life from creatures and monsters who will not rest until they've had you for dinner? Are you looking for a real-time game with a little different theme? Well, look no further. Pack up your cares and woes, gather your family and friends around the computer, and settle in for a few games of *Gravity-Float Trace*.

This is a game of skill and timing that uses the Lo-Res color graphics and the game paddles of the Apple. The computer will draw a different multi-colored line across the playing field each time, and a vertical white line at the right. The object is to trace along that multi-colored line with your "scoring pellet," and reach the white line.

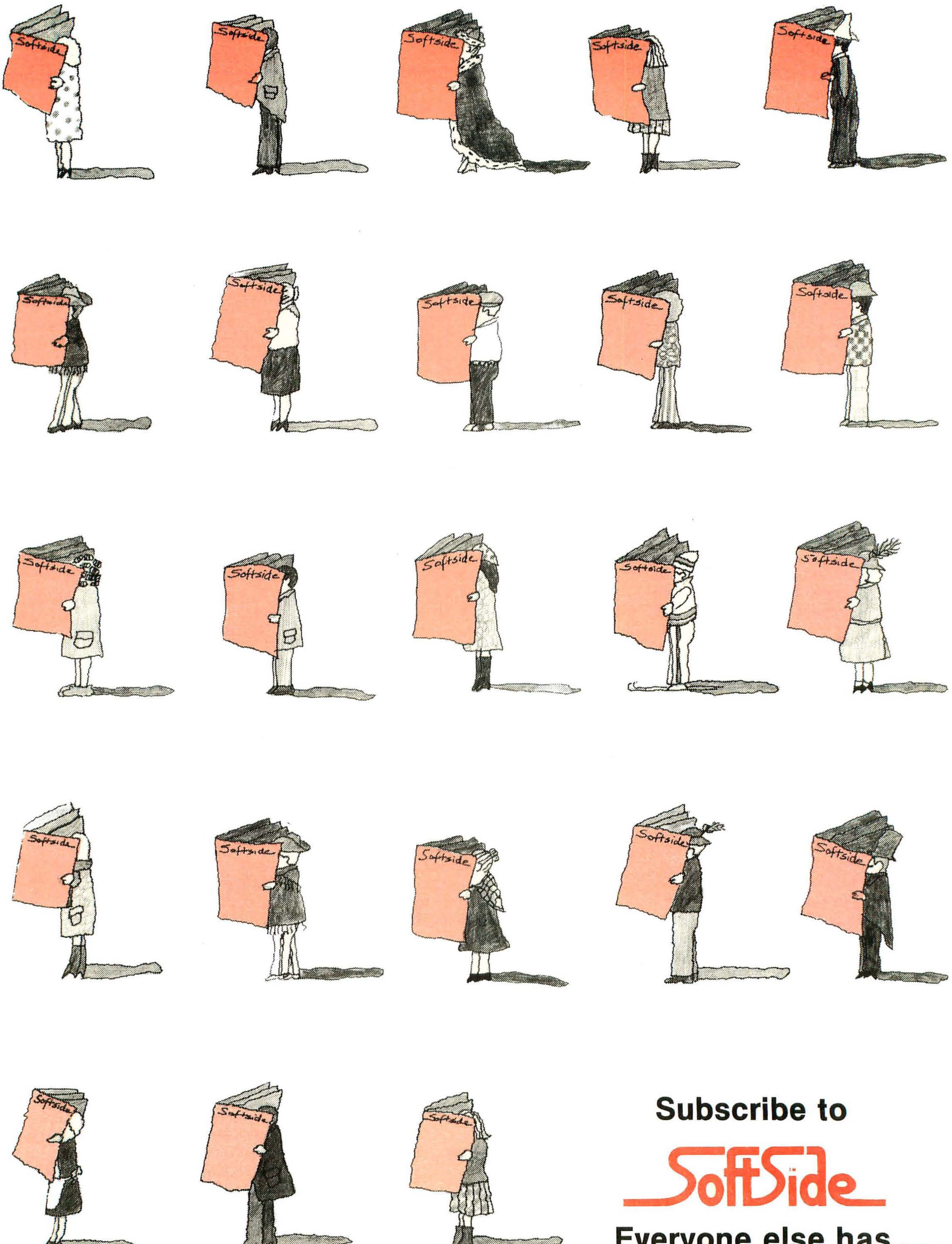
Sound simple? Well, there are a couple of things you should know. There are top and bottom barrier lines running horizontally across the screen; if your scoring pellet hits one of these, you lose all your points. Each time you cause the scoring pellet to "trace" the multi-colored line you score points, depending on the color of the line segment and the skill level you have chosen. There are three skill levels, with the higher ones giving you less time and room in which to make mistakes.

The scoring pellet is controlled by the pushbutton on Paddle 1. After the platform which holds the pellet "ticks" out from under it, gravity begins to pull it down, toward the bottom barrier. At the same time, the pellet begins moving at a constant speed to the right, toward the white finish line. By pressing the button on Paddle 1, you activate an anti-gravity beam which causes the pellet to float upward like a balloon. But don't hold the button too long, or you'll crash into the top barrier. By holding and releasing the button skillfully enough, you can control the movement of the scoring pellet and pick up points along the way.

The program keeps a running tally of points as they are scored. At the end of each game, the final score and the high-game score are both shown. The graphics are, of course, in color; and the game is complete with sound effects.

Variables

BP: Bonus points.
BY: Bottom barrier's y-coordinate.
CC$: Accepts player's answer.
CN: Skill level choice number.
CX: Scoring pellet speed control variable.
CY: Scoring pellet's starting y location.
D$: Waits and receives user input.
DC: Controls direction of plotting, up or down.
KK: Current x-coordinate for plotting pellet.
KT: Color number at point where pellet is going.
LHS: Highest score.
LS: Last score before loss.
NA, NB, NE: Loop variables.
NM$: Player's name.
Q: Random number used for processing.
S: Speaker tweaking address; equals -16336.
SB: Loop variable.
SC: Current score.
SST: Used for timing loop.
TB: Contents of paddle button PEEK address; greater than 127 if pressed.
TY: Top barrier's y-coordinate.
W: Used for timing loop.
XS, YS: Used for x- and y-coordinates for PLOT in scoring curve subroutine.
Y: Used as y-coordinate in removing platform.
Z1, Z2: Random numbers between 0 and 9.
Z3: Always 1, 2, or 3; used for color selection.



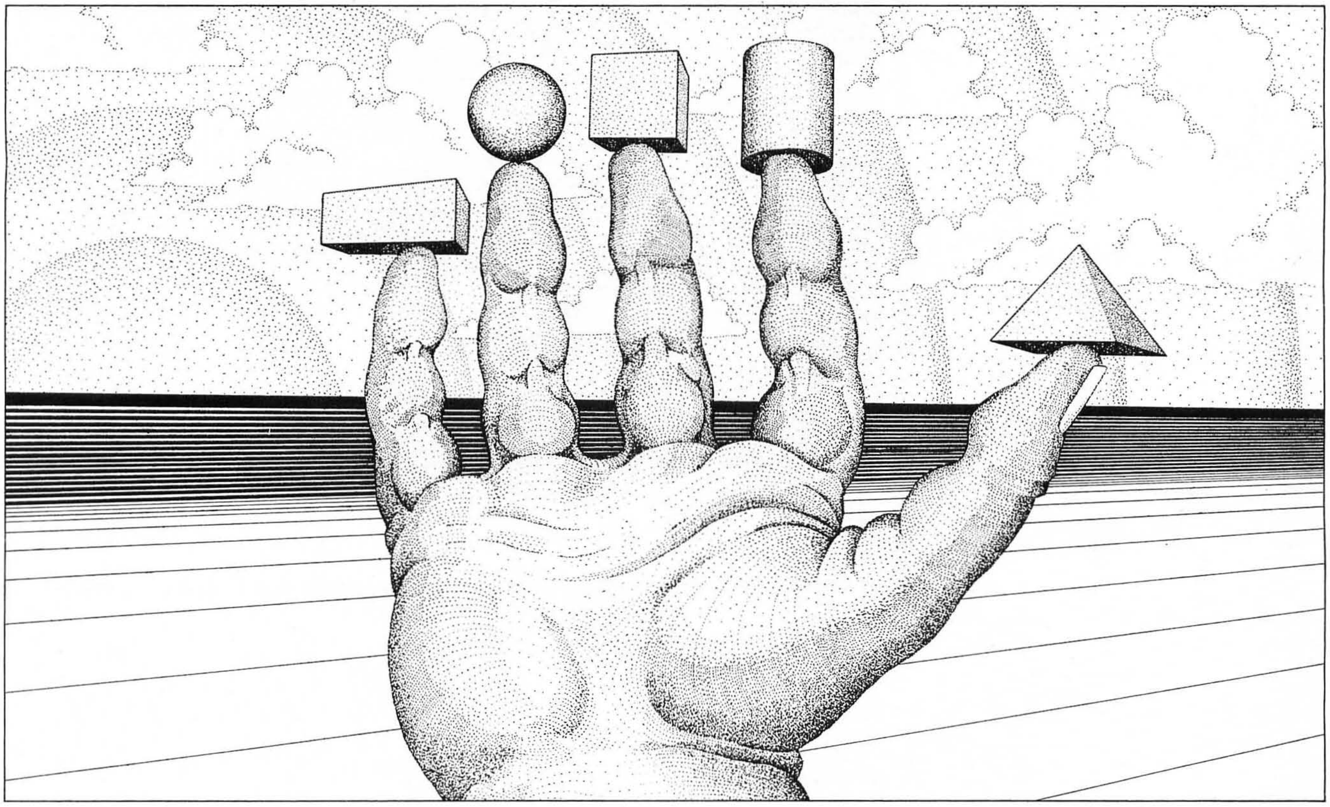
Subscribe to

SoftSide

Everyone else has . . .

See binding card on page 81.

*TRS-80 DIVERSITY AT YOUR FINGERTIPS



AOS EXPANSION PACKAGE 1

There are *two* separate programs included in this expansion package which help you get more out of your *TRS-80. VARKEEP is a variable-passing utility which can SAVE variable values and pass them to other programs, allowing you to perform true program chaining. SCREENPACKER is a powerful graphics utility which allows you to "draw" images, produce large graphic letters, and SAVE them on disk.

This package is available for disk operated systems only, specify Model I or Model III

MACRO-MONITOR . . . THE SHADOW

What secrets lurk deep within the heart of your microprocessor? Only THE SHADOW knows. This amazing machine language program by Jake Commander disassembles and examines program instructions from any part of your computer memory. THE SHADOW also allows you to single-step through your *TRS-80 Model I or III ROM. THE SHADOW will allow you to load a machine language program and begin execution (at a user-specified breakpoint) one instruction at a time with a user-defined time delay, routing it and the current register value to your video screen or printer. THE SHADOW is available on disk and tape for Model I, or disk for Model III.

VOYAGE OF THE VALKYRIE

You are in command of the attack ship Valkyrie. Your mission is to battle your way past giant, laser-wielding war birds and capture the island Fugloy. This AOS adventure game by Leo Christopherson challenges you to choose the proper mountain passes to reach the 10 island castles and secure the gold hidden within. Absolutely flickerless birds sail smoothly across the video screen to strains of Wagner's "Ride of the Valkyries" in what Christopherson considers his best work to date. There are 10 successive levels of difficulty to insure lasting interest and enjoyment.

VOYAGE OF THE VALKYRIE is available for *TRS-80 tape and disk.

Now available at your local software dealer, or call (800) 348-8558 to order. (Indiana residents call (219) 879-4693). MasterCard and VISA accepted.

OTHER ADVANCED OPERATING SYSTEMS PROGRAMS

Circuit Design Group:	**Disk**	**Tape**
Active Filter Design	$39.95	$34.95
Descriptive Statistics and Regressive Analysis	$39.95	$34.95
Electronics I	$39.95	$34.95
Electronics II	$39.95	$34.95
Electronics III	$39.95	$34.95
Plotting Graphs for Line Printer	$39.95	$34.95
Plotting Graphs for Video Display	$39.95	$34.95
EE Ladder Network Analysis	$59.95	$54.95
Macro-Monitor . . . The Shadow Model I	$59.95	$54.95
Model III	$69.95	—
Mostly Basic—Educational	$29.95	$24.95
—Household	$29.95	$24.95
—Scientific	$29.95	$24.95
Time Dungeon American	$29.95	$24.95
Time Dungeon World	$29.95	$24.95
AOS Expansion Package I	$99.95	—
Voyage of the Valkyrie	$39.95	$34.95

ADVANCED OPERATING SYSTEMS
450 St. John Road
Michigan City, IN 46360

*TRS-80 is a registered trademark of Radio Shack, a division of Tandy Corp.

APPLE™

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$    'GRAVITY-FLOAT TRACE'     $
$  AUTHOR: ROBERT PRITCHETT    $
$    (C) 1982    SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization, title, and instructions.

```
10  DEF  FN R(X) =  INT ( RND (1)
     * X)
20  GOSUB 3000
30  PRINT : PRINT : HTAB 2
40  INPUT "WOULD YOU LIKE INSTRUC
     TIONS (Y OR N)?";CC$
50  IF CC$ = "Y" THEN  GOSUB 5000
60  INPUT " WHAT IS YOUR NAME, PL
     EASE? ";NM$
```

Draw new game screen and sound speaker.

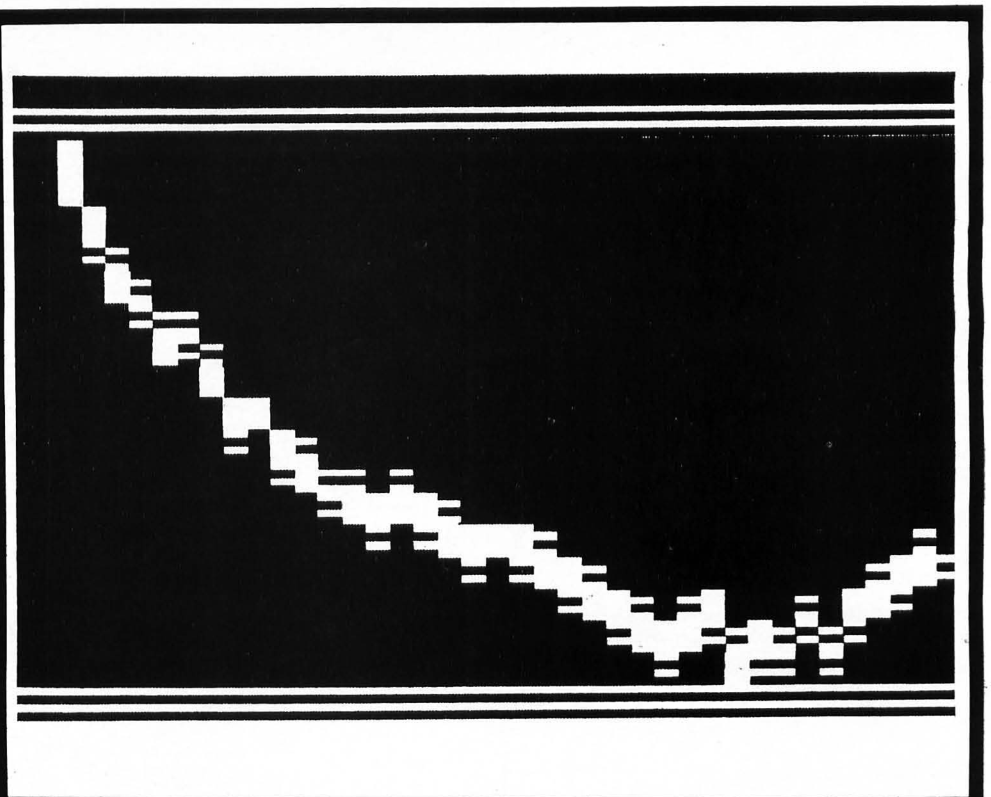
```
70  GOSUB 4400
95  GOSUB 4200
```

Choose skill level.

```
100  PRINT : PRINT "PLEASE CHOOSE
      A SKILL LEVEL (1,2,3)"
110  PRINT "1 EASIEST - 2 HARDER
      - 3 HARDEST"
120  INPUT "ENTER CHOICE NUMBER -
      > ";CN
130 Q =  FN R(10)
140  IF Q = 0 THEN Q = Q + CN
150 BP = CN + CN * Q + Q * 10
155  HOME
160  PRINT "YOU'RE GOING FOR ";BP
      ;" BONUS POINTS"
```

Draw top and bottom barriers.

```
170  COLOR= 0
180  ON CN GOTO 210,220,230
205  GOTO 95
210  HLIN 0,39 AT 2: HLIN 0,39 AT
      38:TY = 2:BY = 38:CY = 4:CX =
      50
215  GOTO 240
220  HLIN 0,39 AT 5: HLIN 0,39 AT
      35:TY = 5:BY = 35:CY = 7:CX =
      25
225  GOTO 240
230  HLIN 0,39 AT 10: HLIN 0,39 AT
      30:TY = 10:BY = 30:CY = 12:C
      X = 0
240 XS = 0
```



Plot scoring curve.

```
250  GOSUB 3300
```

Plot platform.

```
260  COLOR= 0
270  GOSUB 4450
275  PRINT "WHEN SUPPORT PLATFORM
      DISAPPEARS"
280  PRINT "THE SCORING PELLET WI
      LL FALL"
285  INVERSE : PRINT " GOOD LUCK
      - ";NM$;"      ": NORMAL
290  INPUT " PRESS RETURN BUTTON
      TO BEGIN ";CC$
```

Remove platform and begin game.

```
295  HOME : GOSUB 4000
298  COLOR= 0:KK = 1: INVERSE
310  FOR I = 1 TO 3
320 TB =  PEEK ( - 16286)
330  IF TB > 127 THEN 350
340 CY = CY + 1: GOTO 360
350 CY = CY - 1
360 KT =  SCRN( KK,CY)
370  IF KT = 9 THEN SC = SC + 10 *
      CN
380  IF KT = 3 THEN SC = SC + 50 *
      CN
390  IF KT = 11 THEN SC = SC + 25
      * CN
410  IF SC > LS THEN  CALL  - 198
420  VTAB 21: PRINT " -*- SCORE -
      *-  => ";SC;"  <= "
430  VTAB 23: PRINT " POSSIBLE BO
      NUS ADD-ON => ";BP;"  "
450  PLOT KK,CY
460 LS = SC
470  IF CY = TY OR CY = BY THEN 6
      10
480  IF KK = 39 THEN 700
490  FOR W = 1 TO CX: NEXT W
500  NEXT I:KK = KK + 1
510  GOTO 310
```

You lose.

```
610  FOR I = 1 TO 50: GOSUB 4100
620  NEXT I
630  INVERSE
640  GOSUB 3900
650  PRINT " ! ! ! YOU HIT A BARR
      IER ! ! !          "
660  PRINT " YOU LOSE ALL YOUR PO
      INTS -> ";LS;" "
670  PRINT "YOUR SCORE -> 0000  H
      IGH SCORE -> ";LHS;" "
680  NORMAL
690  GOTO 900
```

You made it!

```
700  INVERSE
710  GOSUB 3900
715  GOSUB 4200
720  PRINT " !!! YOU'VE MADE IT T
     O THE GOAL !!!  "
730  PRINT "YOU'VE EARNED A ";BP;
     " POINT BONUS "
740 LS = LS + BP
750  IF LS >  = LHS THEN LHS = LS
760  PRINT " YOUR SCORE -> ";LS;"
      HIGH SCORE -> ";LHS
770  FOR II = 1 TO 3
780  CALL  - 198: FOR SST = 1 TO
     100: NEXT SST
790  NEXT II
```

Play again?

```
800  NORMAL
900  INPUT "ANOTHER GAME ??? (Y/N
     ) ";CC$
910 SC = 0:LS = 0
920  IF CC$ = "N" THEN 940
930  GOTO 70
```

End routine.

```
940  FOR W = 1 TO 3
950  CALL  - 198
960  FOR SST = 1 TO 200: NEXT SST
970  NEXT W
975  TEXT
980  HOME : VTAB 12: HTAB 19
985  PRINT "BYE": GOSUB 4200
990  VTAB 23: END
```

Title page.

```
3000  TEXT : HOME
3002 S =  - 16336
3005  INVERSE
3010  VTAB 7: HTAB 10
3020  PRINT "                    "
3030  HTAB 10
3040  PRINT "                    "
3050  HTAB 10
3060  PRINT "   GRAVITY-FLOAT    "
3070  HTAB 10
3080  PRINT "       TRACE        "
3081  HTAB 10
3082  PRINT "                    "
3084  HTAB 10
3085  PRINT "                    "
3086  NORMAL
3087  PRINT : PRINT : PRINT : PRINT
     : PRINT
3088  PRINT "  (C) THE CREATIVE M
     IND WORKSHOP - 1981"
3092  FOR SST = 1 TO 2000: NEXT S
     ST
3145  HOME
3150  INVERSE : HTAB 13
3154  PRINT "              ": HTAB
     13
3155  PRINT "              ": HTAB
     13
3156  PRINT "    + + +     ": HTAB
     13
3157  PRINT "              ": HTAB
     13
3158  PRINT "   GRAVITY    ": HTAB
     13
3159  PRINT "              ": HTAB
     13
3160  PRINT "    FLOAT     ": HTAB
     13
3161  PRINT "              ": HTAB
     13
3162  PRINT "    TRACE     ": HTAB
     13
3163  PRINT "              ": HTAB
     13
3164  PRINT "    + + +     ": HTAB
     13
3165  PRINT "              ": HTAB
     13
3166  PRINT "              ": HTAB
     13
3167  NORMAL
3168  PRINT : PRINT : PRINT
3169  HTAB 10
3170  PRINT "THE APPLE VERSION 1.
     0"
3171  PRINT : HTAB 19: PRINT "BY"
     : PRINT
3172  HTAB 12: PRINT "ROBERT PRIT
     CHETT "
3240  RETURN
```

Routine to plot scoring curve.

```
3300  GOSUB 3610
3330  GOSUB 3710
3332  GOSUB 3770
3340  IF Z3 = 1 THEN  COLOR= 9
3350  IF Z3 = 2 THEN  COLOR= 3
3360  IF Z3 = 3 THEN  COLOR= 11
3369  IF XS >  = 39 THEN 3550
3370  FOR SLI = 1 TO 2
3375  IF XS >  = 39 THEN 3470
3378 DSP = DC:DSP = YS:DSP = CN:D
     SP = XS
3380  IF YS <  = TY + 1 OR YS >  =
     BY - 1 THEN 3440
3390  PLOT XS,YS
3400  IF DC = 1 THEN YS = YS + 1
3410  IF DC = 2 THEN YS = YS - 1
3420  IF Z1 = 3 OR Z1 = 6 OR Z1 =
     8 THEN XS = XS + 1
3430  GOTO 3470
3440 DC = DC + 1
3450  IF DC = 3 THEN DC = 1
3452  IF YS <  = TY + 1 THEN YS =
     YS + 1
3454  IF YS >  = BY - 1 THEN YS =
     YS - 1
3455  GOSUB 3610
3460  GOTO 3390
3470  NEXT SLI
3480 XS = XS + 1
3490  IF Z2 < 3 AND DC = 1 THEN Y
     S = YS - 1
3495  IF Z2 > 7 AND DC = 2 THEN Y
     S = YS + 1
3500  IF XS >  = 39 THEN 3550
3510  GOSUB 3610
3520  GOTO 3340
3550  RETURN
```

Scramble dice.

```
3610 Z1 =  FN R(10)
3611 Q =  FN R(10)
3613 Z1 = Z1 * 3
3615  IF Z1 > 9 THEN Z1 = Z1 - (Q
     + CN)
3620  IF Z1 <  = 0 THEN Z1 =  FN
     R(CN) + 1
3622  IF Z1 < 10 AND Z1 > 0 THEN
     3625
3623  GOTO 3615
3625 Z2 =  FN R(Z1) * CN + CN
3628 Q =  FN R(10)
3630  IF Z2 > 9 THEN Z2 = Z2 - (Q
     + CN)
3633  IF Z2 <  = 0 THEN Z2 = Z1 +
     1
3635  IF Z2 < 10 AND Z2 > 0 THEN
     3650
3640  GOTO 3628
3650  IF Z1 <  = 3 THEN Z3 = 1
3660  IF Z1 > 3 AND Z2 <  = 6 THEN
     Z3 = 2
3670  IF Z1 > 6 THEN Z3 = 3
3680  RETURN
```

Random y start.

```
3710 YS = Z1 + Z2 + Z3 + CN * 3
3720  IF YS <  = TY + 2 OR YS >  =
     BY - 2 THEN 3735
3730  RETURN
```

Flip coin.

```
3735 Q =  FN R(10)
3740  IF YS <  = TY + 2 THEN YS =
     YS + TY + Q
```

```
3745  IF YS > = BY - 2 THEN YS =
     YS - TY + Q
3750  GOTO 3720
3770  IF Z3 = 3 THEN 3790
3780 DC = Z3: GOTO 3800
3790 DC = 2
3800  RETURN
```

Print blank lines.

```
3900  FOR I = 1 TO 4
3910  PRINT "
                              "
3920  NEXT I
3930  RETURN
```

Remove platform.

```
4000  COLOR= 1
4020  IF CN = 1 THEN Y = 5
4030  IF CN = 2 THEN Y = 8
4040  IF CN = 3 THEN Y = 13
4060  FOR SB = 2 TO 1 STEP  - 1
4070  PLOT SB,Y
4075  GOSUB 4100
4080  FOR SST = 1 TO 900: NEXT SS
     T
4085  NEXT SB
4088  PLOT 0,Y: GOSUB 4100
4095  RETURN
```

Click speaker.

```
4100 SOUND =  PEEK (S) +  PEEK (S
     ) +  PEEK (S)
4110  RETURN
```

Sound up.

```
4200 S =  - 16336
4210  FOR NA = 1 TO 7
4220  FOR NB = 1 TO 10:SOUND =  PEEK
     (S) -  PEEK (S): NEXT NB
4240  FOR NE = 1 TO 7 - NA: NEXT
     NE
4260  NEXT NA
4300  RETURN
```

Draw new game screen.

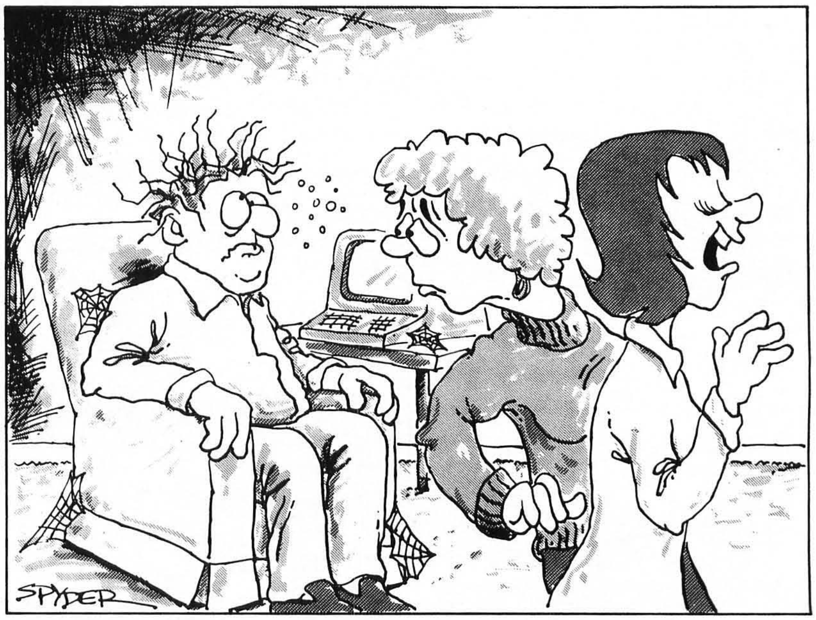
```
4400  GR : COLOR= 15: VLIN 0,39 AT
     39
4405  COLOR= 1
4410  FOR I = 0 TO 38
4415  VLIN 0,39 AT I
4420  NEXT I: RETURN
```

Plot scoring pellet and platform.

```
4450  ON CN GOTO 4470,4480,4490
4470  PLOT 1,4: HLIN 0,2 AT 5
4475  GOTO 4500
4480  PLOT 1,7: HLIN 0,2 AT 8
4485  GOTO 4500
4490  PLOT 1,12: HLIN 0,2 AT 13
4500  RETURN
```

Print instructions.

```
5000  HOME
5010  HTAB 14
5013  HTAB 14: PRINT "INSTRUCTION
     S": PRINT
5020  HTAB 15: PRINT "WELCOME TO"
     : PRINT
5023  HTAB 10: PRINT ":*: GRAVITY
     -FLOAT :*:": PRINT
5024  HTAB 2: PRINT ":*: THE UNIQ
     UE, FUN TRACING GAME :*:": PRINT
5026  PRINT "THE OBJECT OF THE GA
     ME IS TO FOLLOW "
5030  PRINT "AND TRACE THE COLORE
     D LINE WITHOUT"
5033  PRINT "HITTING THE BLACK BA
     RRIERS AT THE TOP "
5036  PRINT "AND BOTTOM OF THE PL
     AYING FIELD, AND TO MAKE IT
     TO THE WHITE LINE AT THE RIG
     HT!"
5040  PRINT "THE SCORING PELLET W
     ILL BE PULLED BY "
5043  PRINT "GRAVITY UNTIL YOU PR
     ESS THE BUTTON ON"
5046  PRINT "PADDLE #1; THEN AN A
     NTI-GRAVITY RAY"
5060  PRINT "CAUSES THE SCORING P
     ELLET TO RISE UNTIL"
5063  PRINT "YOU RELEASE THE BUTT
     ON; THEN GRAVITY "
5070  PRINT "PULLS THE SCORING PE
     LLET AGAIN, ETC.  :*:": PRINT
5072  INPUT " :*: PRESS RETURN FO
     R SCORING INFO :*:";D$
5074  HOME : HTAB 10: PRINT "- **
     * SCORING *** -"
5076  PRINT
5080  PRINT "10 PTS.  - TOUCH ORA
     NGE LINE"
5090  PRINT "50 PTS.  - TOUCH LT.
      PURPLE LINE"
5100  PRINT "25 PTS.  - TOUCH PIN
     K": PRINT
5103  PRINT "ABOVE SCORES ARE MUL
     TIPLIED BY THE "
5105  PRINT "SKILL LEVEL NUMBER C
     HOSEN": PRINT
5110  PRINT "MAKE THE GOAL LINE A
     ND RANDOM BONUS"
5120  PRINT "WILL BE ADDED TO YOU
     R SCORE": PRINT
5130  PRINT "HIT A BARRIER LOSE A
     LL YOUR POINTS    "
5140  PRINT
5150  INPUT "PRESS RETURN BUTTON
     TO PLAY - GOOD LUCK  ";D$
5200  RETURN
```

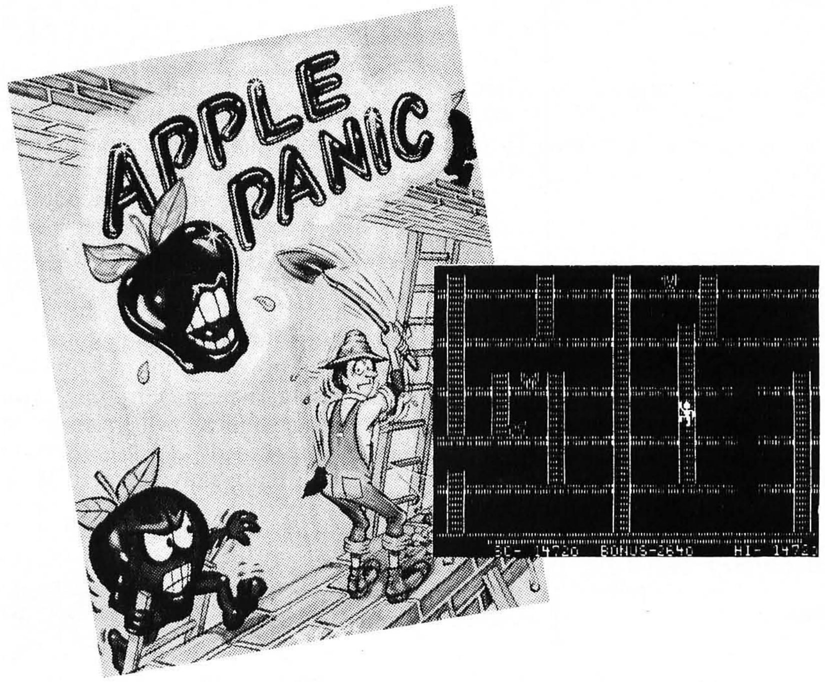


"SO ONE DAY HE FIGURED OUT HOW TO INFINITELY ACCELERATE THE SPEED ON HIS 'SPACE INVADERS' PROGRAM..."

Apple Panic

A review by Hartley G. Lesser

from Broderbund Software. System requirements: 48K Apple II or Apple II Plus with one disk drive, DOS 3.3. Suggested retail price: $29.95.



Beware, one and all. The brilliantly red apple you select in the produce section today may turn and relish **you** on the morrow! Broderbund Software's release of *Apple Panic* does little to nullify such a nightmare, as phobic pippins and their eager allies totally commit themselves to your utter destruction. This game is one of the few that is almost as much fun to watch as it is to participate in, and is addictive to boot.

The setting for this fruit-garnished hunt appears to be the cross-section of a building under construction. Five brick floors are displayed horizontally across the screen, with vertical ladders of varying heights interconnecting some of the levels.

The player is represented in bipedal form, and begins the game at ground level, carrying a pickaxe over one shoulder. Your screen figure is moved by using the "I", "J", "K", and "M" keys which correspond to up, left, right, and down. The pickaxe is primarily used as a tool for digging and filling holes. Its other purpose is as a weapon to be used against the motile fruit.

The only method by which the player can stop the fearless apples is to dig holes, for these renegades from a pie must fall into these traps to be halted. But you must position your character's feet correctly before you can dig a hole in the bricks. This will require practice, and inevitably, many a new game, for the apples wait for no man or woman.

By carefully watching your character run up and down ladders, and across the levels, the day will finally come when you will instinctively know when to press the "A" key. If you have timed the key-press correctly, your apple attacker will halt and commence to swing the pickaxe into the bricks, causing a hole to appear. It is into this hole the enemy must fall. Don't be satisfied with a single floor opening. Dig many, and on all levels. Watch the apples, and you may discern a pattern to their movements, which may assist you with placement of your traps.

Success in trapping an apple is only the first step. To soundly mash the MacIntosh, you must calmly walk up to the holed creature and press the "S" key. This activates the pickaxe which beats mercilessly upon the fruit's felonious head. This will ultimately cause the apple to fall through the hole, which destroys the turnover-turncoat and gives the player 100 points.

If the player has lined up several holes, one beneath the other, each level the apple tumbles is worth an additional 100 points. However, lest this seem a simple task, rest assured there will be more than one apple enemy to confront. As the player betters his or her score, the number of the cider brethren increases as well.

The letter "S" is also used on those occasions when you may feel it necessary to fill in a previously dug opening. This may happen if you find yourself trapped on a level with an advancing Granny Smith before you. To your rear, there is a ladder to safety, but a hole blocks your retreat. You have two choices.

Your character could jump through the hole. This action incurs no damage whatsoever to your MacIntosh-masher. However this would not be feasible if doing so would bring you face to face with other apples on the floor below.

The other recommended action would be to fill in the opening, which is accomplished by using the letter "S" when standing next to the hole. But you need time to do that and if the chomping, stomping, Granny Smith is too near, then by all means jump!

The apples are not the player's only concern. After annihilating the first two offerings of apples, a deadlier menace will appear. This is the infamous Green Butterfly, a winged creature of death that aids the apples in their hunt for you. This particular opponent must be sent crashing down at least two levels, so the traps must be dug exactly one under the other. Should you fail in this regard, all of your digging will have been for naught. The butterfly will merely tumble one level, brush the dust from its gossamer wings, and once again take up the chase. Should you succeed in eliminating the Green Butterfly, your reward is 200 points per level fallen.

The other ally of the forbidding fruit is the Mask of Utter and Total Horror and Death. This morbid character requires three floors of descent for destruction, with an appropriate 300 points per level fallen. Fair warning — this is no easy matter to accomplish.

As the stalwart apple beater, you begin the game with three chances, or "lives." You will gain an extra life upon winning the confrontation with the Green Butterfly. Your life status is continually displayed in the lower left-hand corner of the screen.

Upon the demise of an apple, the Butterfly, or the Mask, you will note with some annoyance that the holes through which they have descended no longer exist. This is calculated to bring tears to the most ardent of gamers, for there is but one thing left to do. Redig those holes if you feel they were strategically significant to your success.

Each time you clear the screen of the Golden Delicious and their sidekicks, bonus points are earned. The rapidity of this accomplishment will also garner the gamer extra points. For new players, however, survival, regardless of the length of time taken, will be of foremost concern.

A striking offense is the best defense against these mad and impetuous apples, and a quickly learned alacrity in climbing and descending ladders will also hold one in good stead.

What initially may seem to be a trivial or elementary arcade game will soon prove its worth as one of the most played in your software library. I highly recommend *Apple Panic* . . . core, stem, seeds and all.

amp recordings

duplicating service

307 West Main Street
Maple Shade, NJ 08052

(609) 667-1667

- AMP "Data-sette" blank cassettes for digital use
- Cassette Storage Boxes
- Cassette Labels - Custom printing & blank
- Custom Record Album production from your tapes
- Stereo and Spoken Word cassette duplication

Call or write to:

Jerry

for more information.

All cassette work at AMP R. & D. is custom work to fit your needs.

SPORTS FANS!

MASTERS' GOLF

by David Bohlke

Winter's almost over, but there's at least two feet of snow on the ground. Lee Trevino is out on a golf course, but you're not. You could follow the sun in pursuit of your favorite pastime, but that can be quite expensive. . .

Don't despair! Power up your TRS-80® or ATARI® and prepare to tee off. The multi-talented Dave Bohlke has created this golf program to help you get over those missing links blues. For 1-4 players, **Masters' Golf** (TRS-80® version published originally as **Protour Golf**) provides all the excitement of a nine-hole round of golf, including full club selection, sand traps, and contoured greens. The money you save on caddy fees is enough to pay for endless rounds of golf played right in your own home.

TRS-80® 16K Tape $9.95
ATARI® 8K Tape $9.95

WORLD SERIES

by David Bohlke

Ah yes, it's almost spring and a young man's thoughts turn to. . . baseball? How would you like a baseball season with no threats of player strikes or free agent negotiations that leave the shattered remnants of once-mighty teams strewn about the playing fields? We offer you your own league. . . Apple, ATARI® or TRS-80®. Batter up!

TRS-80® 16K Tape $9.95
Apple 16K Tape $9.95
Apple 32K Disk $14.95
ATARI® 16K Tape $9.95

For ordering information, see page 10.

RAMWARE

SoftSide Selections
6 South Street Milford NH 03055

COMMANDing BASIC

An Ampersand Tutorial

by Michael Prescott

No doubt Ampersand (&) is one of the most obscure and mysterious functions in Applesoft. Maybe you've read about its use before, but never fully understood how to use it yourself. The only mention of it in the Applesoft manual is on page 123, and gives a rather intimidating impression to the reader. True, Ampersand is not for the beginner, but then, a beginner would not have the need to pass variables directly to a Machine Language program. In fact, Ampersand is not much more difficult to use than a CALL or USR. The only difference is that the routine called by the & must be made able to interpret all the characters that follow the "&".

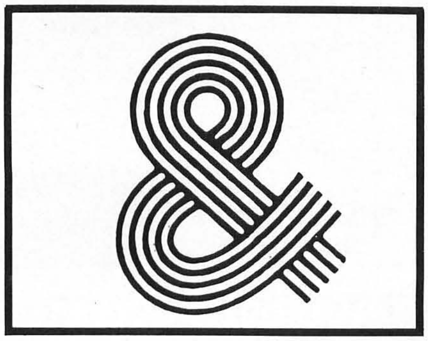
From here on, we will have to make some assumptions about your knowledge of Machine Language. We will assume you already have a working Machine Language program named "A", a basic understanding of Machine Language commands, and a method of entering Machine Language into memory.

To start with, when the Apple II finds an "&" in either deferred execution (i.e., a running program) or immediate execution (i.e., typing on the keyboard), the system does a CALL 1013. It's as simple as that.

At this point we will enter the monitor 6502 language. In monitor, "1013" is equivalent to $03F5. The $ symbol indicates that we are using base 16 or hexadecimal (0-F), instead of base 10 or decimal (0-9). At $03F5 the Apple expects to be told where to go. Thus $03F5 should be a jump instruction, JMP, to the beginning of the routine to fetch the variables for routine "A", let's assume $0300 or 768 base 10. This is where the April, 1980, ***SoftSide*** article on Ampersand left off. It simply put a JMP command in $03F5.

Next, it is necessary to understand how Applesoft organizes the characters in a program or input line. First, all spaces are removed except those between quotes. Next, all reserved words are detected and reduced to a single character. So actually, a program line can be thought of as a single block of characters, and that the LIST command was made to automatically insert spaces for easier reading.

The Apple system interprets a program line one character at a time (since reserved words occupy only one character) through the use of a text pointer, whose function is similar to running your finger across a page as you read it. The first character of a program line segment must be a reserved word, to tell the Apple what to do, followed by any other characters which conform to proper Apple syntax. The program segment ends with a ":" or end-of-line character. If these rules are not followed, the text pointer will point to something the Apple has no idea what to do with, thus giving you a SYNTAX ERROR.



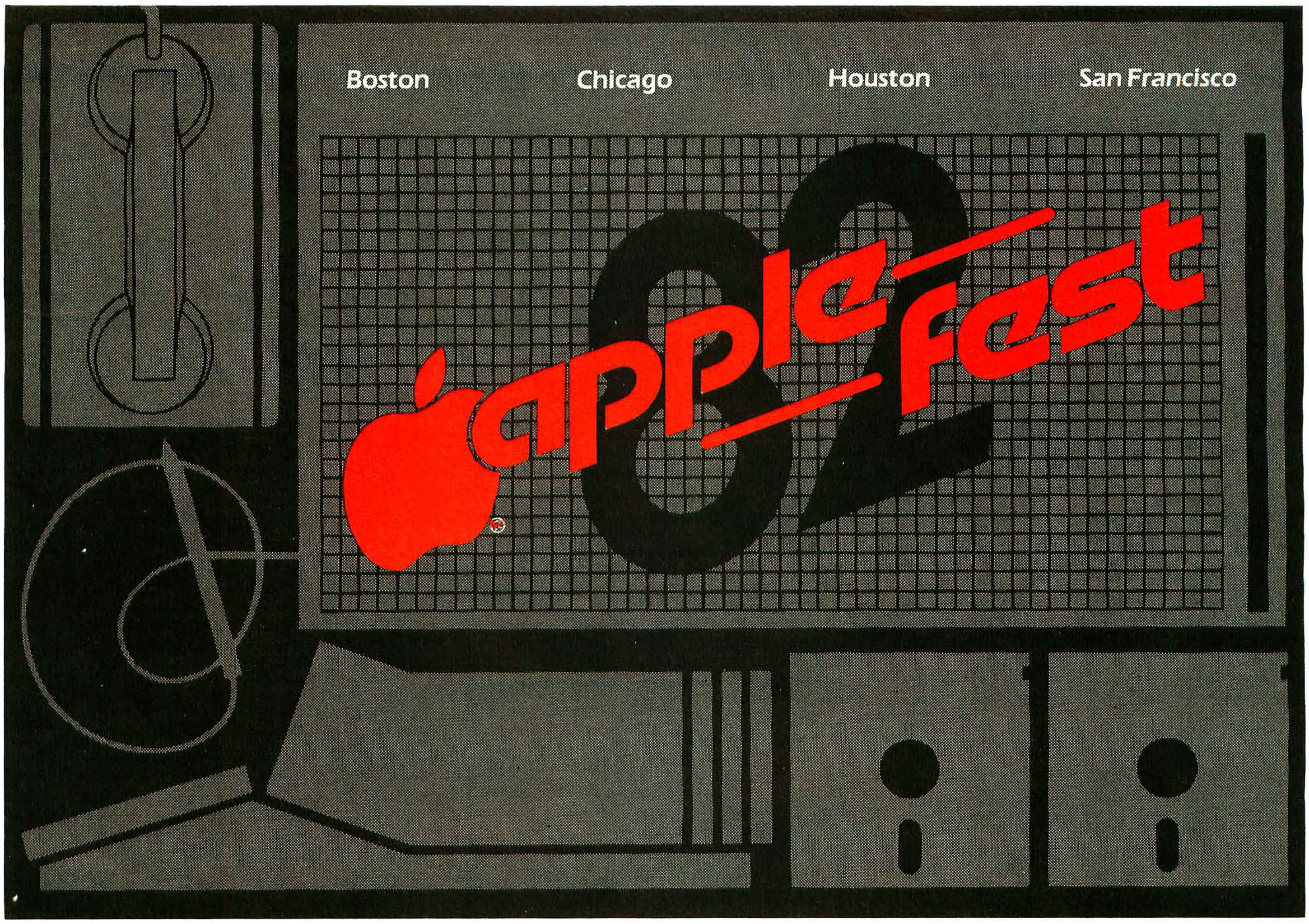
Upon detecting an "&" the Apple system will do a CALL 1013 as described earlier, but will leave the text pointer pointing to the character immediately following the "&". So it is up to your Machine Language variable fetching routine to interpret all the characters up to the next ":" or end-of-line character. This isn't as hard as it sounds, as you will soon see.

The characters pointed to by the text pointer will be placed in the A-register. From here you may test the character for a special value (e.g., &PLOT 23,15) to vector the routine to a special subsection of the variable fetching routine. To do this use a CMP# value and BEQ address command. This is similar to the BASIC statement "IF A-register = value THEN...". If you use a special character you must move the text pointer over it before reading the value that follows it. To do this we'll use a routine called CHRGET located at $00B1. In order to do this, you must input a JSR$ 00B1, which is similar to the BASIC GOSUB. CHRGET advances the text pointer by one character and loads the A-register with the ASCII value of the next character.

For simplicity, we will only consider transferring numeric variables using the &. Suppose that we now wish to read in the value that is pointed to by the text pointer. To do this use a JSR to the Floating point INput routine, FIN, located at $EC4A. This will cause the text pointer to move along, building a number, until it reaches the first non-numeric character; this is similar to the VAL function in BASIC. The number input will lie in what is called a Floating point ACcumulator, FAC, which occupies $009D-00A2. However, the FAC is in a floating point format which is difficult to handle directly through Machine Language.

To remedy this we could do a JSR to a subroutine called AYINT, located at $E10C, which converts FAC to an integer provided that FAC is greater than -32768 and less than 32768. The integer would lie in $00A0-00A1 where $00A0 = FAC MOD 256 and $00A1 = INT (FAC/256). If this were the only value to be passed into routine "A" then we could directly jump (JMP) to it and the routine should work properly, returning to Applesoft.

Suppose, however, that we have more than one value to be interpreted. As described earlier, the FIN routine moves along getting a number until it stops on a non-numeric character. Thus, multiple values must be separated by something such as a comma. You must next skip over this character using a JSR to CHRGET,



The Most Spectacular Extravaganza Ever... For Apple Users

At Applefest '82 hundreds of manufacturers, distributors and dealers will showcase the entire spectrum of Apple-compatible products including computers, components, peripherals, plug-in cards, publications, gifts, magazines, services, accessories and software for home, office and school.

Hands-on centers and multimedia presentations will demonstrate the newest applications for business, education and entertainment.

Seminars and workshops, conducted by the world's leading Apple authorities, will detail new uses to make your Apple more enjoyable and more useful than you ever imagined.

You'll meet thousands of other Apple owners and find the newest of everything for your Apple under one roof . . . and for sale at super show prices.

So if you use an Apple . . . or are thinking about buying one, you won't want to miss a minute of Applefest '82.

Ticket & Hotel Information

Send your check and a note indicating the specific show you wish to attend. Tickets and hotel information will be mailed back to you. Tickets can also be purchased at the show. Make all checks payable to Northeast Expositions Inc. 824 Boylston Street, Chestnut Hill, Mass. 02167 Tel: 617 739 2000.

Applefest/Boston
Fri-Sun May 14-16, 1982
Hynes Auditorium
Show Hours: 11AM to 6PM Daily
Admission: $6 per day or $10 for 2 days, $15 for 3 days

Applefest/Chicago
Fri-Sun Nov 5-7, 1982
Arlington Park Racetrack/Exposition Center
Arlington Heights, Ill.
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest/Houston
Fri-Sun Nov 19-21, 1982
Albert Thomas Convention Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest/San Francisco
Fri-Sun Dec 3-5, 1982
Moscone Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

SYNCHR, or CHKCOM. The routine SYNCHR starts at $DEC0 and simply verifies that the character pointed to by the text-pointer matches the character of the A-register. If they match, the routine exits via CHRGET; if not, you get a SYNTAX ERROR. The CHKCOM routine starts at $DEBE and simply loads the A-register with the ASCII value of a comma and exits via SYNCHR.

This is fine for values, but what about variables, expressions, or even functions? Luckily, the Apple routines also include a routine called FRMNUM starting at $DD67. FRMNUM evaluates the formula pointed to by the text pointer and does whatever is necessary to get a result in FAC. This routine works with values as well as expressions, and so may be substituted for the FIN routine. However, as in FIN, the routine will stop on a comma or other such separator.

So the overall process is to use FRMNUM or FIN, to get a number; AYINT, to convert it to an integer; then to save the integer; and if there is more than one input then to skip the separating character and repeat the process. There's got to be a better way, and there is!

If you are dealing with integers greater than or equal to 0 and less than 256 then you may use the GETBYT, GTBYTC, or COMBYTE routines. The GETBYT routine resides at $E6F8 and simply evaluates the expression pointed to by the text pointer, converts it to an integer and leaves the result in the X-register. How convenient! The GTBYTC routine starts at $E6F5 and does a CHRGET, skipping a character, and exits via GETBYT. This is useful when you wish the routine to skip a data separator. COMBYTE at $E74C checks for a comma at the textpointer, does a CHRGET and exits via GETBYT.

The following are the formal definitions of routines mentioned, useful conversion functions, and references for further study.

CHRGET $00B1
Increment textpointer, load A-register from text pointer.

FIN $EC4A
Input a floating point number into FAC from CHRGET. FIN assumes that the 6502 registers have been set by the CHRGET that fetched the first digit.

AYINT $E10C
If FAC is less than 32767 and greater than -32767 then leave INT(FAC) in FACHO, MO,LO signed.

SYNCHR $DEC0
Checks at text pointer for the character in A-register. Normally exits through CHRGET. Exits with SYNTAX ERROR if they don't match.

CHKCOM $DEBE
Checks at text pointer for "," uses SYNCHR.

FRMNUM $DD67
Evaluate the formula at text pointer, put it in FAC, and make sure it's a number. On entry text pointer points to the first character of the formula. TYPE MISMATCH ERROR results if the formula is a string.

GETBYT $E6F8
Evaluates the formula at text pointer, leaves the result in FAC, and converts FAC into a single byte number in X-register and FACLO. Normally exits through CHRGET. If FAC is greater than 255 or less than 0 then exits via ILLEGAL QUANTITY ERROR.

GTBYTC $E6F5
JSR to CHRGET and fall into GETBYT.

COMBYTE $E74C
Check for a comma and get a byte in X-register. Uses CHKCOM, GETBYT on entry text pointer points to comma.

To convert base 16 to base 10:
A$= xxxx, where xxxx is greater than or equal to 0000 and less than or equal to FFFF,len(A$)=4.

```
10 A=0:FORX = 1 to 4: A=A +
16  (4-X)* (ASC(MID$(A$,X,1)) -
48 -7 *(MID$(A$.X,1)  "9")):
NEXT: PRINT A
```

To convert base 10 to base 16:
A=xxxx, where xxxx is greater than or equal to 0 and is less than or equal to 65336.

```
10 FOR X = 3 TO 0 STEP-1:
PRINT CHR$(48 + INT(A/16  X)
+ 7 * (A/16  X  9));: IF
A  16  X THEN A = A -
16  X*INT(A/16  X)
20 NEXT
```

Hot pursuit through space and the vortices of time!

Time Lord

The fallen Time Lord, who presumptuously calls himself The Master, is at large.

The elders of Waldrom have supplied you with the hyperspace-worthy vessel Tardis, and commissioned you to eliminate the evil "Master." Your resources include clones who will fight for you, the formidable CRASER weapons of the Tardis, and magic weapons such as Fusion Grenades and Borelian Matrix Crystals.

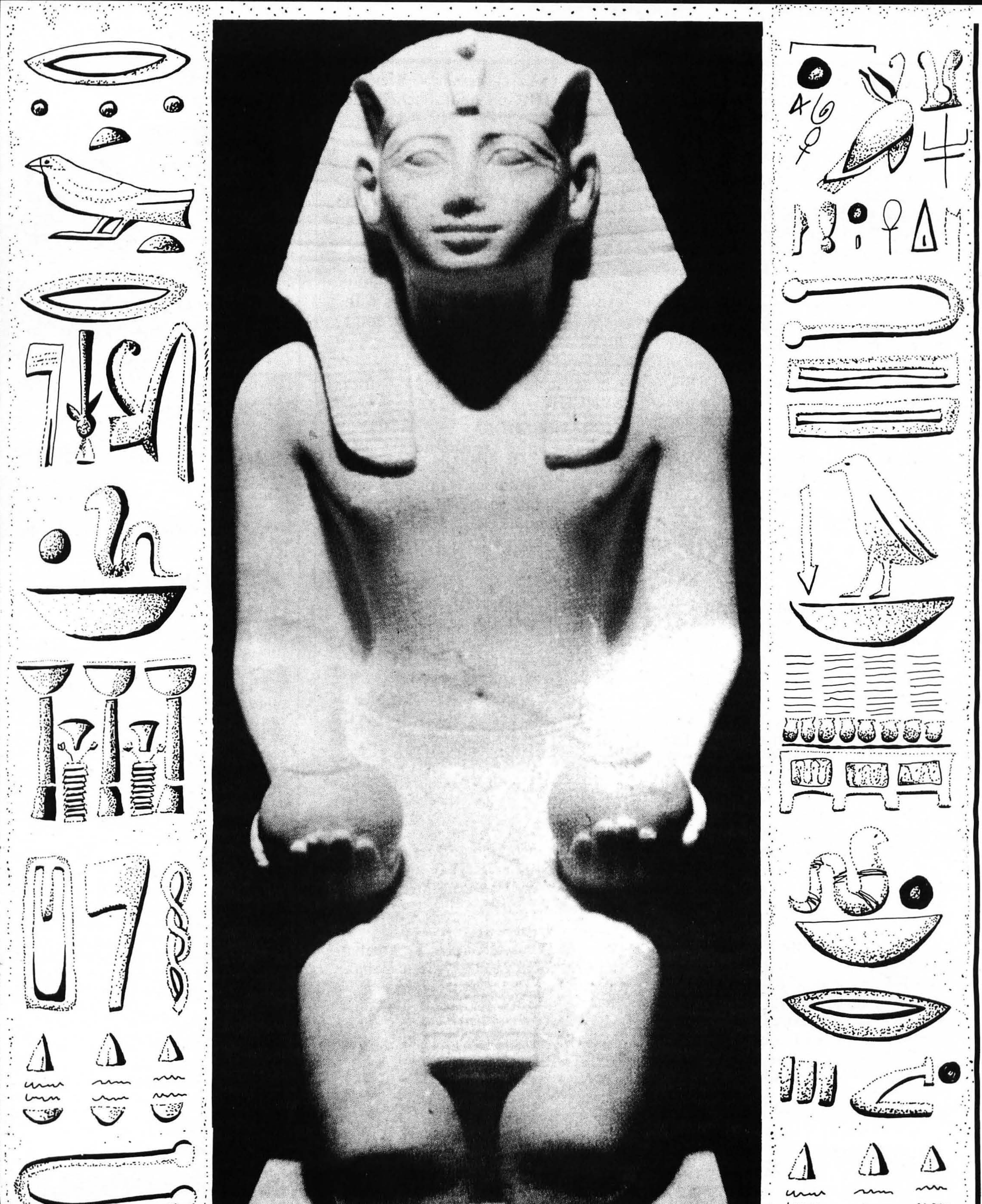
Traveling through hyperspace in search of the evil one, you will encounter Time Eaters, Neutron Storms, and other alien creatures and phenomena. Entering real space to search planets, you will encounter still other dangers. You will enter native settlements to buy food and supplies — or to fight for survival.

Once you find The Master can you destroy him?

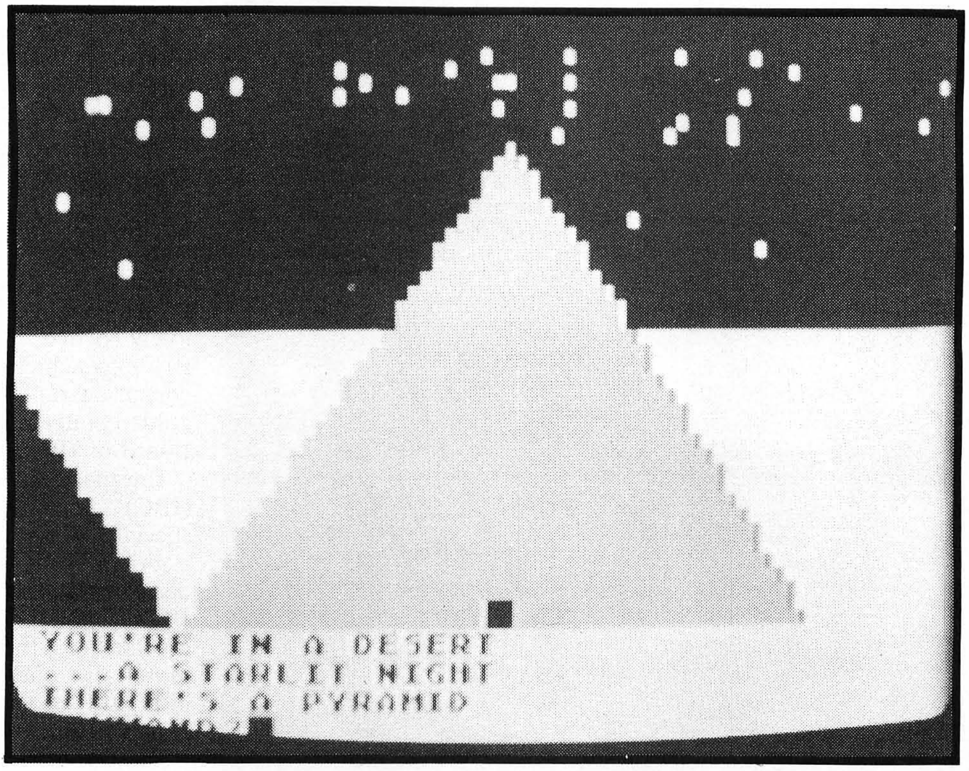
Apple Integer BASIC
Disk 48K $29.95

For ordering information, see page 10.

SoftSide Selections
6 South Street Milford NH 03055



THE Curse OF THE PHARAOH



```
YOU'RE IN A DESERT
...A STARLIT NIGHT
THERE'S A PYRAMID
```

by Peter Kirsch

***The Curse of the Pharaoh* is a graphics adventure for an ATARI® with disk drive and 32K RAM. It is included as a bonus program on this month's ATARI® Disk Version.**

Many, many years ago a lone thief defiantly entered the pyramid tomb of the Pharaoh Ickabathan and stole the two rubies which had been placed as the eyes of the mummy. The thief, however, became confused and hopelessly lost trying to escape this tomb of death, and in the process dropped the rubies. Sightless, the mummy sprang to life and instinctively sought out the intruder, viciously slaying him on a staircase. The mummy returned to its sarcophagus, but placed a curse of darkness upon the land until its eyes should be returned.

Years have passed. You will attempt to end the curse by recovering the lost rubies and returning them to their rightful owner. There are two: one red and one green. The red ruby, although hidden, should be relatively easy to find. The green one is another matter altogether. In order to succeed you must find both, return them to the mummy, and exit the pyramid.

This adventure has a total of 22 locations, each room graphically displayed in three dimensions. As you enter each room you will be facing north. All exits, except any to the south (behind you), will be clearly indicated on the walls. To travel in a given direction, simply type N, E, S, W, U, or D. To take inventory of what you are carrying, type I.

Instead of typing a full two-word command as in most adventures, all you need to do is to type the keyword: GET or OPEN, for example, instead of GET BOX or OPEN DOOR. And, if an item is not relevant in a particular situation, you will not be able to DROP an item you are carrying. Such memory-conserving devices allow the game to fit into just 16K of RAM.

Many of the rooms are drawn using routines and subroutines from other rooms (such as left or right walls). After you solve the adventure and want to examine the listing, you can notice the use of the XIO command to fill an area between four plotted points or lines with a given color. This is used to color the pyramid, the desert, and several of the rooms and other features inside.

Entering an illegal command will result in a loud buzz from the computer. Legal actions are confirmed by a bell tone.

Good luck on your perilous mission!

INTERNATIONAL BRIDGE CONTRACTORS

by Phillip Case

ATARI® version by Patrick Maloney, translation contest winner for March

International Bridge Contractors **is a business management simulation for an ATARI® 400/800 with 16K RAM or 24K RAM with disk.**

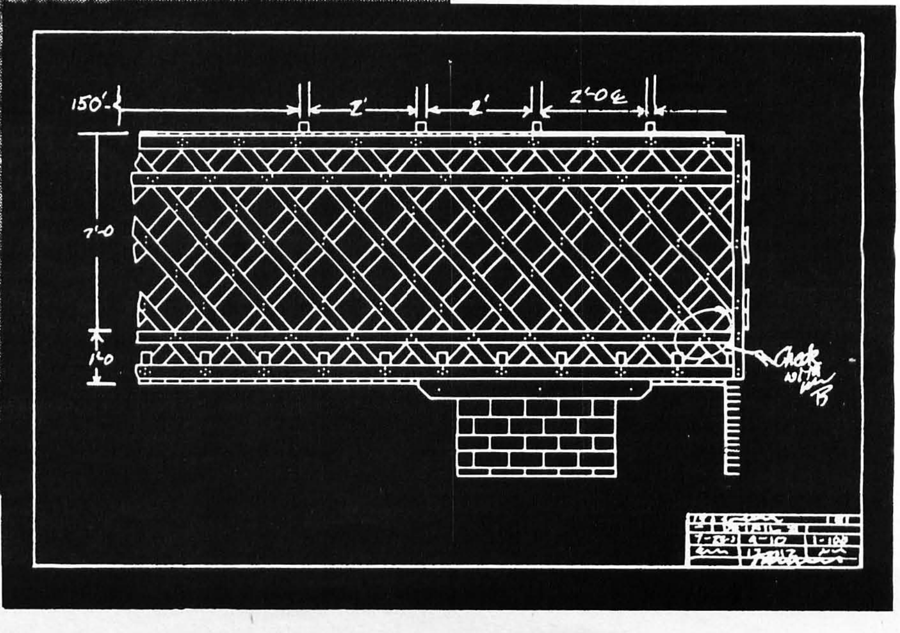
As you read the company newsletter, you can't believe your eyes. The chairman of the board is on vacation in Bermuda. While he's basking in the warm tropical sun, you're freezing to death building bridges in below zero temperatures, surrounded by the snow-covered New Hampshire wilderness.

Your goals are set, but you wonder if you can attain them. From your present position as a lowly office manager, you must strive to achieve and advance your corporate rank until you reach the golden plateau of "Chairman of the Board of IBC, Inc."

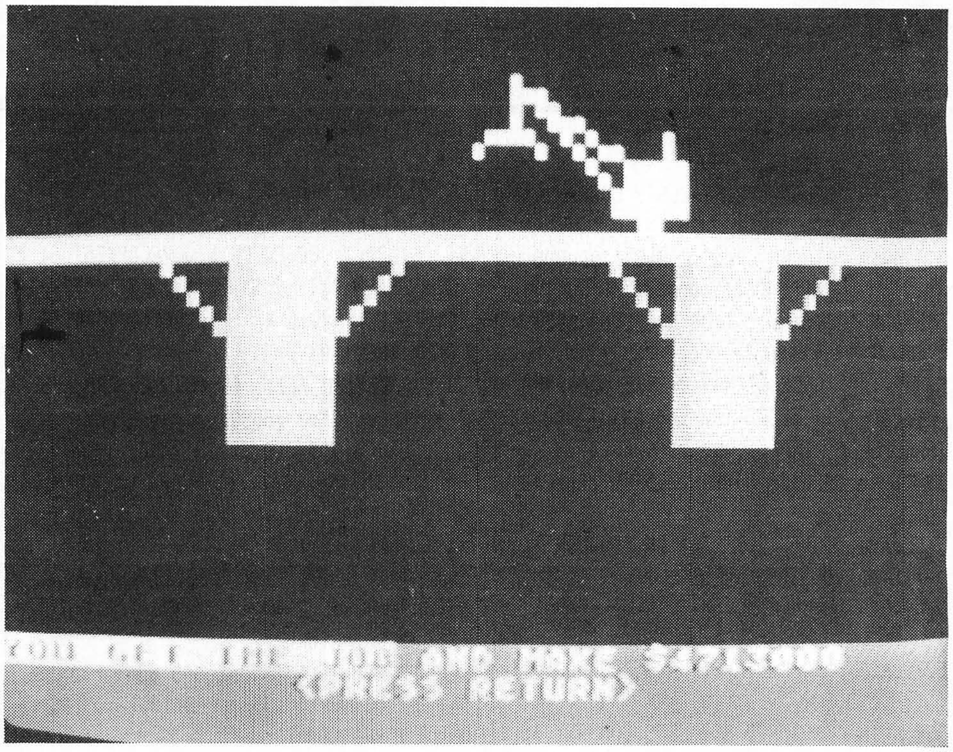
International Bridge Contractors (IBC) is a corporation which specializes in bridge construction on a global scale. You must make the decisions governing how successful IBC becomes, from hiring of work crews, to purchasing building materials and making contract bids to prospective clients for construction.

Can you handle the awesome responsibilities of running a large international corporation? Or will you go bankrupt and spend next week in the lines at the unemployment office?

Find your answers in *IBC*.



ATARI®


YOU GET THE JOB AND MAKE $4713000
<PRESS RETURN>

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          ATARI BASIC         $
$  INT'L BRIDGE CONTRACTOR     $
$     AUTHOR: Phillip Case     $
$  TRANSL: Patrick Maloney     $
$       (c) 1982 SoftSide      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

5 OPEN #1,4,0,"K:"
10 DIM NA$(8),P$(5),B$(9),T$(6),A$(6),
C$(5),S$(5),R$(25),STAR$(40),DA$(103),
X$(6),CL$(1)
20 STAR$="*****************************
***********":CL$=CHR$(125)
25 DIM MST$(17)
30 MST$="*****************"
100 GRAPHICS 2:POKE 752,1
102 POKE 82,0
104 ? "  International  Bridge  Contra
ctors"
105 ? "           By William Phillip Cas
e      "
106 ? "  Atari translation by Patrick
Maloney ";
108 POKE 82,2
110 FOR Z=1 TO 400:POSITION 2,4:? #6;"
******IBC******":NEXT Z
120 ? CL$;" International  Bridge  Con
tractors"
130 GOSUB 30010
150 ? " PLEASE ENTER YOUR NAME";:INPUT
 NA$:C=40000000:P$="LIGHT":PC=75000:PS
=3000:PM=5000:B$="LIGHT-MED":BC=210000
160 GOSUB 30010
170 ? :GOSUB 2000:GOSUB 30010
250 GRAPHICS 0:POKE 752,1:? CL$
251 SETCOLOR 1,0,4:SETCOLOR 2,0,12
257 BS=100:BM=150:T$="MEDIUM":TC=30000
0:TS=1000:TM=2000:A$="MEDIUM"
260 AC=345000:AS=1200:AM=2200:C$="HEAV
Y":CC=525000:CS=2500:CM=3500:S$="HEAVY
":SC=998000:SS=3000:SM=6000
261 R$="OFFICE MANAGER"
270 TI=TI+1
280 GOSUB 300:GOSUB 400:GOSUB 500:GOSU
B 3000:GOSUB 4000:GOSUB 5000:GOSUB 550
0:GOSUB 6000
290 GOTO 270
300 POKE 82,1:? CL$;"NAME:";NA$:? "POS
ITION:";R$:? STAR$:? "COMPANY STATUS D
ISPLAY       TURN:";TI:? STAR$
305 ? "  BRIDGE STATUS"
310 POSITION 4,6:? #6;"TYPE:";:POSITIO
N 12,6:? #6;"COMPLETE";:POSITION 4,8:?
 #6;"PONTOON";:POSITION 4,9
311 ? #6;"SGL BEAM"
```

```
315 POSITION 4,10:? #6;"TRUSS";:POSITI
ON 4,11:? #6;"ARCH";:POSITION 4,12:? #
6;"CANTILVR";:POSITION 4,13
317 ? #6;"SUSPEN"
320 POSITION 14,8:? #6;B1;:POSITION 14
,9:? #6;B2:POSITION 14,10:? #6;B3:POSI
TION 14,11:? #6;B4;
325 POSITION 14,12:? #6;B5:POSITION 14
,13:? #6;B6;
330 POSITION 21,5:? #6;"COMPANY ASSETS
:":POSITION 20,6:? #6;MST$:POSITION 21
,7:? #6;"WORK CREWS=";CR:POSITION 21,8
333 ? #6;"R&D=$";RD:POSITION 21,9:? #6
;"STEEL=$";M
335 POSITION 21,10:? #6;:POSITION 21,1
1:? #6;"CASH=$";C
340 POSITION 10,18:? #6;"<PRESS RETURN
>";
341 GET #1,XV
350 GOSUB 30010:RETURN
400 GOSUB 30020:? CL$;"NAME:";NA$:? "P
OSITION:";R$:? STAR$:? "COMPANY PURCHA
SES DISPLAY      TURN:";TI:? STAR$:?
403 ? "1. HIRE 5 WORK CREWS-$1000000"
405 ? "2. PURCHASE STEEL----$2000000"
410 ? "3. R&D INVESTMENT----$5000000":
? :? STAR$:? "  CASH =$";C:? STAR$:? "
ENTER CHOICE ('0'TO CONTINUE)";
415 TRAP 400:INPUT CH:IF CH<0 OR CH>3
THEN 400
416 IF CH=0 THEN GOSUB 30010:TRAP 4000
0:RETURN
420 ON CH GOSUB 430,440,450
425 GOSUB 30020:GOTO 400
```

```
430 CR=CR+5:C=C-1000000:RETURN
440 M=M+2000000:C=C-2000000:RETURN
450 RD=RD+5000000:C=C-5000000:RETURN
500 TRAP 500:RESTORE :? CL$;"YOUR SECR
ETARY REPORTS:":X=INT(38*RND(1))+1:? S
TAR$
501 IF X<11 THEN ? "NOTHING IMPORTANT
IS IN THE NEWS.":? STAR$:? :GOSUB 2000
:RETURN
502 X=X-10:FOR RK=1 TO X:READ DA$,MP:N
EXT RK:? DA$:IF X<15 THEN C=C-MP:GOTO
504
503 C=C+MP
504 ? STAR$:? "PRESENT CASH =$";C:? ST
AR$:? :GOSUB 2000:TRAP 40000:RETURN
511 DATA "NEW TARIFF COSTS COMPANY $10
00000.",1000000
512 DATA "ACCOUNTING ERROR COSTS COMPA
NY         $10000.",10000
513 DATA "YOUR BOSS JUST ORDERED YOU T
O SPEND      $5000 FOR A WILDLIFE PRESER
VE.",5000
514 DATA "FORMER EMPLOYEE COLLECTS $10
0000        INSURANCE CLAIM AGAINST THE
COMPANY.",100000
515 DATA "WORKER SLOWDOWN COSTS THE CO
MPANY      $3000.",3000
516 DATA "CONDEMNED OFFICE COSTS COMPA
NY $300000 TO REBUILD.",300000
517 DATA "NEW TRADE LAWS COST COMPANY
$2000.",2000
518 DATA "VANDALIZED OFFICE COSTS COMP
ANY $500    IN DAMAGES.",500
519 DATA "WORKER CARELESSNESS COSTS CO
```

```
MPANY      $1000.",1000
520 DATA "LABOR STRIKE COSTS COMPANY $
50000.",50000
521 DATA "STOLEN CRANE COSTS $1000000
TO          REPLACE.",1000000
522 DATA "COMMUNICATION PROBLEMS COST
COMPANY     $7000.",7000
523 DATA "THE CHAIRMAN OF THE BOARD HA
S DIED     IT COSTS THE COMPANY $30000
TO REPLACE HIM.",30000
524 DATA "OFFICE FIRE COSTS COMPANY $3
00000.",300000
525 DATA "GOVT. COMPETITION SHOULDN'T
AFFECT     BUSINESS.",0
526 DATA "FORMER CLIENT GIVES COMPANY
$20000.",20000
527 DATA "NEW CONTRACT SAVES COMPANY $
50000.",50000
528 DATA "CAPTURED SPY GETS COMPANY $5
0000       IN REWARDS.",50000
529 DATA "COMPANY SAVES $2000000 ON FO
REIGN      EQUIPMENT PURCHASE.",2000000
530 DATA "NEW INVENTION GETS COMPANY $
1000000.",1000000
531 DATA "ACCOUNTING ERROR GAINS COMPA
NY $3000.",3000
532 DATA "NEW LAW SAVES COMPANY $10000
.",10000
533 DATA "COMPANY OWNED STOCKS EARN $8
00000      IN DIVIDENDS.",800000
534 DATA "NEW ACCOUNTING SYSTEM SAVES
COMPANY    $4000.",4000
535 DATA "COMPANY GAINS $50000 IN REVE
NUES.",50000
536 DATA "NEW ATARI COMPUTER SYSTEM SA
VES        COMPANY $50000.",50000
537 DATA "PROTEST OF COMPANY'S PRESENC
E IN THE  EVERGLADES DOESN'T AFFECT BU
SINESS.",0
538 DATA "GOVERNMENT INVESTIGATION INT
O ALLEGED UNDERGROUND ACTIVITIES BY CO
MPANY      DOESN'T AFFECT BUSINESS.",0
2000 ? "             <PRESS RETURN>";:GE
T #1,XV
2001 GOSUB 30010:RETURN
3000 ? CL$;"NAME:";NA$:? "POSITION:";R
$:? STAR$:? "RESEARCH & DEVELOPMENT ST
ATUS  TURN:";TI:? STAR$:? :?
3010 X=INT(RND(0)*30)+1:X=X*1000000:IF
 RD>X THEN 3020
3015 POSITION 7,7:? #6;"NO R&D PROGRES
S THIS TURN.":? :GOSUB 2000:RETURN
3020 GH=RD/5000000:SB=INT(RND(0)*GH)+1
3021 X=INT(RND(0)*6)+1:? "R&D REPORTS
A COST BREAKTHROUGH:":? :? "COST OF A
";
3022 ON X GOSUB 3023,3024,3025,3026,30
27,3028:GOSUB 25000:GOSUB 2000:RD=RD-S
B*5000000:RETURN
3023 ? "PONTOON BRIDGE HAS GONE FROM":
? PC;" TO ";PC*0.75;" PER 100/FT.":PC=
PC*0.75:RETURN
3024 ? "SINGLE BEAM BRIDGE HAS GONE FR
OM":? BC;" TO ";BC*0.75;" PER 100/FT."
:BC=BC*0.75:RETURN
3025 ? "TRUSS BRIDGE HAS GONE FROM":?
TC;" TO ";TC*0.75;" PER 100/FT.":TC=TC
*0.75:RETURN
3026 ? "ARCH BRIDGE HAS GONE FROM":? A
C;" TO ";AC*0.75;" PER 100/FT.":AC=AC*
0.75:RETURN
3027 ? "CANTILEVER BRIDGE HAS GONE
  FROM ";CC;" TO ";CC*0.75;" PER 100/F
T.":CC=CC*0.75:RETURN
3028 ? "SUSPENSION BRIDGE HAS GONE
  FROM ";SC;" TO ";SC*0.75;" PER 100/F
T.":SC=SC*0.75:RETURN
4000 ? CL$;"NAME:";NA$:? "POSITION:";R
$:? STAR$:? "PRESENT BRIDGE SPECS DISP
LAY   TURN:";TI:? STAR$
4002 ? "TYPE:";" TRAFFIC:";" PER 100/F
T.";" SAFE--MAX.":? STAR$
4010 ? "PNTOON ";P$;"       ";PC;"    "
;PS;"  ";PM
4011 ? "SGL BM ";B$;" ";BC;"      ";BS;
"   ";BM
4012 ? "TRUSS  ";T$;"      ";TC;"     ";T
S;"  ";TM
4013 ? "ARCH   ";A$;"      ";AC;"     ";A
S;"  ";AM
4014 ? "CNTLVR ";C$;"       ";CC;"     ";
CS;"  ";CM
4015 ? "SUSPEN ";S$;"       ";SC;"     ";
SS;"  ";SM
4016 ? STAR$:? "CASH =$";C;"  STEEL =$
";M:? STAR$
4020 GOSUB 2000:RETURN
5000 ? CL$;"NAME:";NA$:? "POSITION:";R
$:? STAR$:? "CONTRACT BIDDING PHASE
      TURN:";TI:? STAR$
5001 ? "CLIENT'S BRIDGE HAS THESE SPEC
S.":? STAR$
5002 JJ=INT(RND(0)*6)+1:ON JJ GOTO 500
3,5004,5005,5006,5007,5008
5003 X$="PNTOON":GOTO 5009
5004 X$="SGL BM":GOTO 5009
5005 X$="TRUSS ":GOTO 5009
5006 X$="ARCH  ":GOTO 5009
5007 X$="CNTLVR":GOTO 5009
5008 X$="SUSPEN":GOTO 5009
5009 IF JJ=1 THEN Y=INT(RND(0)*PM)+1:Q
=1:X=75000:K=5:F=PC
5010 IF JJ=2 THEN Y=INT(RND(0)*BM)+1:Q
=2:X=210000:K=8:F=BC
5011 IF JJ=3 THEN Y=INT(RND(0)*TM)+1:Q
=3:X=300000:K=15:F=TC
5012 IF JJ=4 THEN Y=INT(RND(0)*AM)+1:Q
=4:X=345000:K=25:F=AC
5013 IF JJ=5 THEN Y=INT(RND(0)*CM)+1:Q
=5:X=525000:K=34:F=CC
5014 IF JJ=6 THEN Y=INT(RND(0)*SM)+1:Q
=6:X=998000:K=50:F=SC
5016 ? "   TYPE = ";X$;"       LENGTH =
";Y;" FT.":? STAR$:? "YOUR COST FOR TH
IS BRIDGE IS $";Y*F/100:? STAR$
5018 IF CR<(Y/100)*Q THEN ? "YOU DON'T
 HAVE ENOUGH CREWS FOR THIS   BRIDGE."
:? :GOSUB 20000:RETURN
5020 IF M<(Y/100)*(K*10000) THEN ? "YO
U DON'T HAVE ENOUGH STEEL FOR THIS   J
OB.":? :GOSUB 20000:RETURN
5025 TRAP 5025:? "HOW MUCH DO YOU WANT
 TO BID";:INPUT BD:? STAR$
5026 TRAP 0
5027 Q=INT(RND(0)*11)+1:IF BD>(Q+1)*((
X*Y/100)*0.9) THEN ? "BID TOO HIGH,YOU
 LOSE CONTRACT":? :GOSUB 20000:RETURN
5030 GOSUB 28000:? "YOU GET THE JOB AN
D MAKE $";BD-(F*Y/100):C=C+BD-(F*Y/100
):M=M-(Y/100)*(K*10000)
5033 IF M<0 THEN M=0
```

Continental Adventures

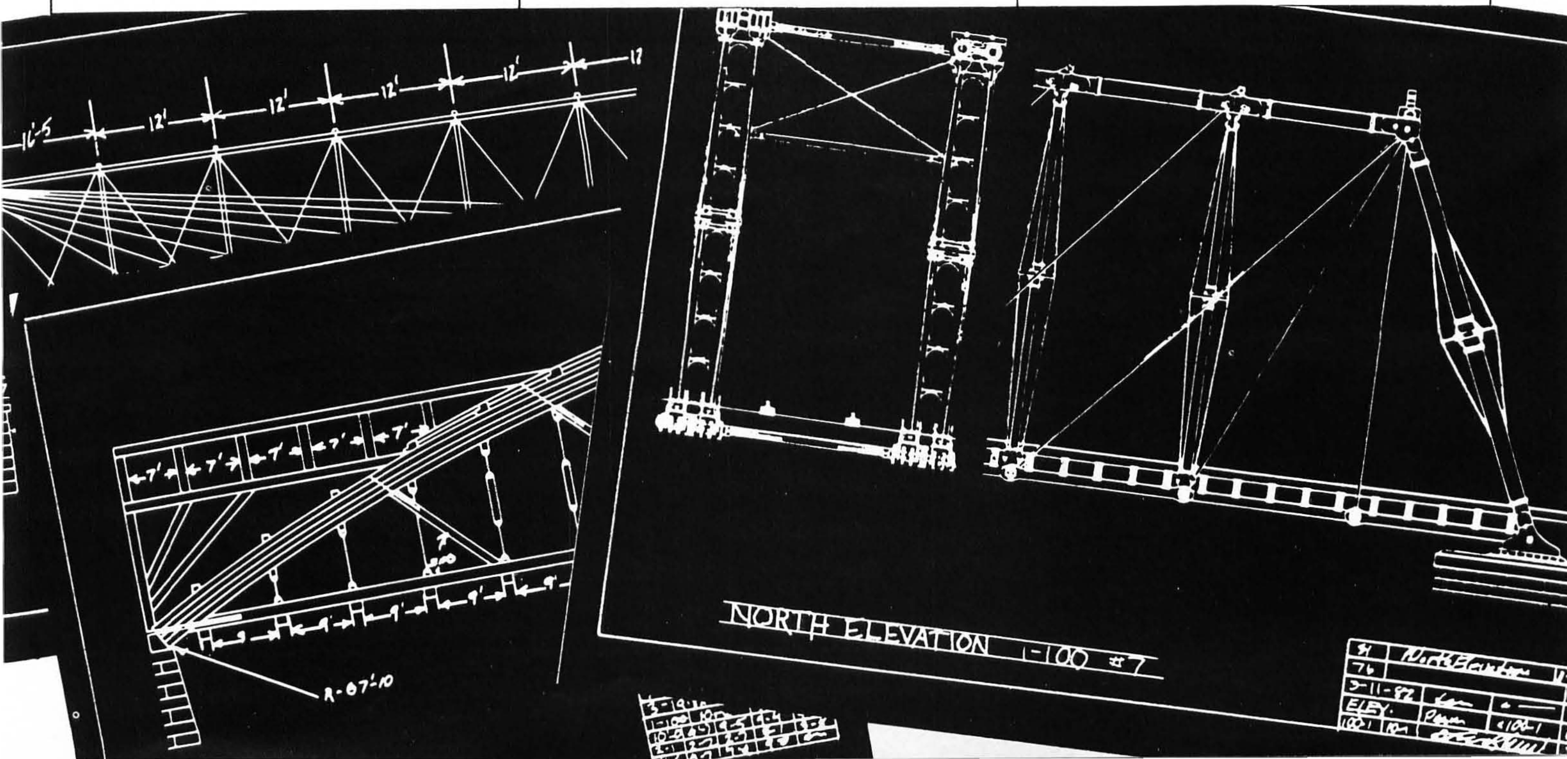
4975 Brookdale Dept. 05
Bloomfield Hills, Mich. 48013
(313) 645-2140

Continental Adventures presents three adventures and one graphics game for the Atari 400 and 800 computer owner

The Ghost Tower — Combat with diabolical demons, 16K $16.95
Town of Derango — Avenging the death of a father, 8K $16.95
Talisman of Power — A search for the four keys of Gremlock, 16K.... $18.95
Super Shape Builder — A graphics game for creating your own pictures. Joysticks reqd. 8K $14.95

ATARI®

```
5035 GOSUB 25000
5040 IF JJ=1 THEN B1=B1+1
5041 IF JJ=2 THEN B2=B2+1
5042 IF JJ=3 THEN B3=B3+1
5043 IF JJ=4 THEN B4=B4+1
5044 IF JJ=5 THEN B5=B5+1
5045 IF JJ=6 THEN B6=B6+1
5050 GOSUB 2000:GRAPHICS 0:POKE 752,1:
SETCOLOR 2,0,14:SETCOLOR 1,9,9:RETURN
5500 X=INT(RND(0)*300)+1:IF C>=0 THEN
RETURN
5505 C=C*1.5:IF C<X*1000000*-1 THEN GO
SUB 9000
6000 IF C>50000000 AND CB<5 THEN R$="D
ISTRICT MANAGER":CB=5
6010 IF C>100000000 AND CB<10 AND CB>4
 THEN R$="REGIONAL SUPERVISOR":CB=10
6020 IF C>150000000 AND CB<15 AND CB>9
 THEN R$="CORPORATE ADVISOR":CB=15
6030 IF C>200000000 AND CB<20 AND CB>1
4 THEN R$="COMPANY REPRESENTATIVE":CB=
20
6040 IF C>300000000 AND CB<30 AND CB>1
9 THEN R$="V.P. OF PRODUCTION":CB=30
6050 IF C>400000000 AND CB<40 AND CB>2
9 THEN R$="COMPANY PRESIDENT":CB=40
6060 IF CB=50 THEN 7000
6070 IF C>500000000 AND CB=40 THEN CB=
50:RETURN
6080 RETURN
7000 ? CL$;"          ";NA$;" HAS BEEN
DECLARED"
7010 ? "          CHAIRMAN OF THE BOARD
                    (GAME OVER)":GOSU
B 27000:RUN
9000 ? CL$," UNFORTUNATE NEWS:"
9001 ? ,"================="
9002 ? "          ";NA$;" HAS GONE BANK
RUPT"
9010 ? "                (GAME OVER)":? :?
 :GOSUB 26000:GOSUB 2000:RUN
20000 FOR X=150 TO 255:SOUND 0,X,10,15
:NEXT X:FOR X=1 TO 100:SOUND 0,255,10,
15:NEXT X
20010 GOSUB 2000:RETURN
25000 FOR X=150 TO 45 STEP -1:SOUND 0,
X,10,15:NEXT X:FOR X=1 TO 100:SOUND 0,
45,10,15:NEXT X:RETURN
26000 FOR X=0 TO 230:FOR ZA=0 TO 3:SOU
ND ZA,X,10,15:NEXT ZA:NEXT X:SOUND 0,2
55,10,15
26010 FOR S=1 TO 200:NEXT S:FOR Z=0 TO
 3:SOUND Z,0,0,0:NEXT Z:RETURN
27000 FOR X=230 TO 25 STEP -1:FOR ZA=0
 TO 3:SOUND ZA,X,10,15:NEXT ZA:NEXT X:
SOUND 0,255,10,15
27010 FOR S=1 TO 200:NEXT S:FOR Z=0 TO
 3:SOUND Z,0,0,0:NEXT Z:RETURN
28000 GRAPHICS 5:POKE 752,1:COLOR 3:FO
R TR=19 TO 26:PLOT TR,15:DRAWTO TR,26:
NEXT TR
28010 FOR TR=53 TO 60:PLOT TR,15:DRAWT
O TR,26:NEXT TR
28020 FOR TR=0 TO 32:PLOT TR,13:DRAWTO
 TR,14:NEXT TR
28025 FOR TR=47 TO 79:PLOT TR,13:DRAWT
O TR,14:NEXT TR
28030 PLOT 19,20:DRAWTO 13,14:PLOT 26,
20:DRAWTO 32,14:PLOT 53,20:DRAWTO 47,1
4:PLOT 60,20:DRAWTO 66,14
28060 COLOR 1:PLOT 52,6:PLOT 52,7:PLOT
 50,12:PLOT 51,12:PLOT 48,10:DRAWTO 40
,2:PLOT 48,11:PLOT 49,8:DRAWTO 40,2
28070 FOR TR=49 TO 53:PLOT TR,8:DRAWTO
 TR,11:NEXT TR
28080 A=3:B=1:X=9:GOSUB 29000:FOR TR=1
 TO 50:NEXT TR:A=0:B=0:X=9:GOSUB 29000
:A=3:B=1:X=10:GOSUB 29000:A=0:B=0:X=10
28090 GOSUB 29000:A=3:B=1:X=11:GOSUB 2
9000:COLOR 1:PLOT 48,10:DRAWTO 40,2:A=
0:B=0:X=11:GOSUB 29000
28100 A=3:B=1:X=12:GOSUB 29000:A=0:B=0
:X=12:GOSUB 29000:A=3:B=1:X=13:GOSUB 2
9000:A=0:B=0:X=13:GOSUB 29000
28110 A=3:B=1:X=14:GOSUB 29000
28120 B=0:X=14:GOSUB 29010:B=1:X=13:GO
SUB 29010:B=0:X=13:GOSUB 29010:B=1:X=1
2:GOSUB 29010:B=0:X=12:GOSUB 29010
28130 B=1:X=11:GOSUB 29010:B=0:X=11:GO
SUB 29010:B=1:X=10:GOSUB 29010:B=0:X=1
0:GOSUB 29010:B=1:A=0:X=9:GOSUB 29010
28140 RETURN
29000 COLOR A:PLOT 33,X-1:DRAWTO 46,X-
1:PLOT 33,X:DRAWTO 46,X
29010 COLOR B:PLOT 37,X-2:DRAWTO 40,X-
4:PLOT 42,X-2:DRAWTO 40,X-4:PLOT 40,X-
3:DRAWTO 40,3
29015 SOUND 0,255,6,15:FOR TR=1 TO 4:N
EXT TR:SOUND 0,0,0,0
29020 RETURN
30010 FOR U=1 TO 4:SOUND 0,20,10,8:GOS
UB 30220:SOUND 0,0,0,0:NEXT U:RETURN
30020 U=INT(RND(0)*100)+50:SOUND 0,U,1
0,8:GOSUB 30220:SOUND 0,0,0,0:RETURN
30220 FOR TT=1 TO 5:NEXT TT:RETURN
```



Let PASCAL-80 talk some sense into your computer

Phelps Gates, the author of "APL-80", brings you "Pascal-80" for your TRS-80®. Now you can add another dimension to your programming skills by using this fast version of the compiled language Pascal.

"Pascal-80" is a powerful, structured and well-defined language for the TRS-80® microcomputer. This easy-to-use language makes writing well-structured, and therefore easily understandable programs simple. "Pascal-80" supports most of the features of UCSD Pascal, including RECORD, SET (to 256 members), FILE (text and record oriented), n-dimensional ARRAY (and ARRAY of ARRAY, etc.), global GOTO, ELSE in CASE statements, and BCD arithmetic accurate to a full 14 places (including log and trig functions), 6-digit optional. "Pascal-80" features a 23600 byte workspace in 48K, a 1000 line per minute compiler, an easy-to-use text editor, and plain English error messages, all the features you would expect in a Pascal costing hundreds more.

Variable Types:	Boolean, integer, char, real, real6, and text.
Constants:	Maxint, minint, true, false, and pi.
Files:	Input, output, and lp.
Procedures:.	Read, readin, write, writein, reset, rewrite, close, seek, cls, and poke.
Functions:	Abs, arctan, call, chr, cos, eof, eoin, exp, inkey, in, mem, odd, ord, peek, pred, round, sin, signif, sqr, sqrt, succ, and trunc.

"Pascal-80" does not implement variant records, pointer and window variables, or functions and procedures used as parameters.
TRS-80® 32K Disk $99.95

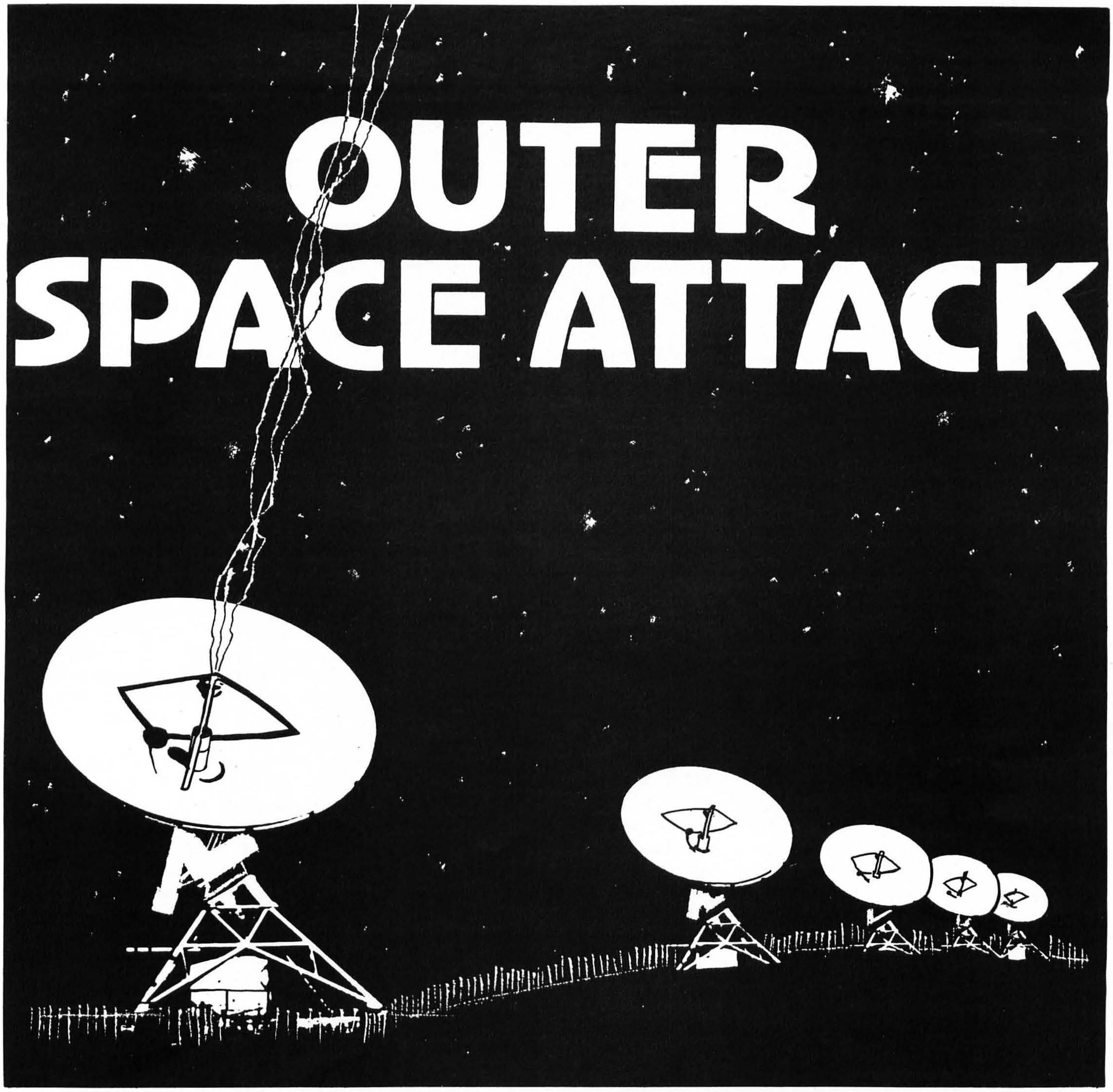
MasterCard VISA

RAMWARE™

For ordering information, see page 10.

SoftSide™ Selections
6 South Street Milford NH 03055

OUTER SPACE ATTACK



by Sheldon Leemon

Outer Space Attack **is a real-time arcade game for ATARI® 400 or 800 computers. It requires 16K ROM (24K for disk), a joystick, and the BASIC cartridge.**

For far too long ATARI® owners have been fed a diet of warmed-over software — programs originally developed for their microcomputer cousins and translated extremely literally for the ATARI®. Although a few Machine Language programs have appeared which take advantage of the graphics features of the ATARI®, BASIC programs which show off the brilliance of the machine are few and far between.

The program presented here is pure ATARI®. Sure, zillions of *Invaders*-type games have been written for microcomputers. But it is precisely because of its wide implementation that I chose this type of program to show off the power of ATARI® BASIC. On what other micro could you write any kind of a playable *Invaders* game in under 50 lines of BASIC? How about one that featured high-resolution graphics, seven colors on the screen at once, sound, and animation?

I don't think we will see any takers for a BASIC translation of this program. No other computer has Player/Missile (PM) graphics, redefinable character sets, relocatable display memory, and the rest of the graphics goodies that come with the ATARI®.

Now, if I were able to do all of this with just a short BASIC program, think what you could do with a long, elaborate one. As more ATARI® owners become aware of these capabilities, we should see innovative, original software designed to utilize the full power of the machine. Until then, I hope you enjoy this newest additon to the *Invaders* horde.

Instructions

Unless you really have been stationed on a lunar defense base, little need be said about how the game is played: You have to shoot down all of the aliens before they get past the horizon (gray surface) and wipe out your planet. To do this, you maneuver your laser base left or right with a joystick in Port 1 and fire with the trigger button.

If you let one slip by, the game will end with the current invasion wave. If you manage to wipe out one wave, a new one will start as soon as the warning siren sounds. To make things worse, the alien menace shoots back, dropping bombs more and more frequently with each successive attack wave. If your laser base is bombed, it will be replaced by one of your two spares, but when they are gone, the game is over.

In the end, your little outpost is doomed. The best you can hope for is to make the aliens pay dearly with their losses (and maximize your score).

Variables

Numeric:

BF, SF: Flags which are set when a shell is travelling upwards or a bomb is travelling downwards.
COUNT, ATTACK: ATTACK holds the number of loop iterations which must pass before a bomb is dropped. COUNT keeps track of them, and is compared to ATTACK to see if it's time to drop one yet.
D: Holds the direction value which tells whether the invaders are moving to the right or left.
DH, DL, L, H: DH and DL hold the location of the two-byte pointer to display memory. L and H hold the values contained in these locations. When new values are POKEd into the pointer, the computer thinks the display starts farther back in memory, and the whole display appears to move.
HITS, CHANCES, SCREENS, SCORE: HITS keeps track of the number of invaders downed on the current assault wave, while CHANCES counts the number of laser bases used up. Each time HITS increases, it adds to the SCORE. After all the invaders are hit, SCREENS is incremented and we start again.
HP, HPM, BVP, SVP: HP holds the horizontal position of the laser base, while HPM does the same thing for the missiles it fires. BVP keeps track of the vertical position of the bombs, and SVP does the same for the shells.
I, J, K: Loop variables. I and J control main program loop that moves the invaders across the screen, and then down at the end of a row.
RT: Holds the value of the top-of-RAM pointer. Used to create a safe storage area above the apparent top of memory.
TOUGH, TOUGHER: Offsets added to the joystick-checking loop. To increase the difficulty in later rounds, part of the loop is skipped. This moves your shells less often, which slows them down, and speeds up the invaders.
X, Y: Hold the x, y location of the invader hit by the last shell.

String:

D$, F$, PM$: These strings are dummies which contain no useful data. They are dimensioned to allow the strings which hold data for the ATARI® PM graphics system to begin on an even 1K boundary.
M$, P$, P1$, P2$: Contain the data for the Players and Missiles which display as the laser bases, bombs, and shells.
BL$: Contains all zeroes; used to clear out other strings.
BOMB$, SHELL$: Hold the shape data for the bombs and shells.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          ATARI  BASIC        $
$     'OUTER SPACE ATTACK'     $
$    AUTHOR: Sheldon Leemon    $
$      (c) 1982  SoftSide      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Lines 30-110: Check the joystick, and move the laser base, missiles, and bombs.

```
10 I=0:J=0:COUNT=0:ATTACK=0:GOSUB 280:
GOTO 140
20 REM -OUTER SPACE ATTACK V.1 12/81
        By: SHELDON LEEMON
        Oak Park, MI 48237
```

Checks the collision register for Missile 0 to see if a missile has hit an invader. If it has, flow jumps to Line 210, where the hit is scored, the invader erased, and a check is done to see if the screen has been cleared.

```
30 Y=PEEK(53248):IF Y>0 THEN GOSUB 210
```

Checks to see if the shell flag (SF) has been set. If it has, the missile is moved, and a check is done to see if it has gotten past the top row of invaders. If it has, the missile is erased and the flag is turned off. Line 50 then repeats the check of Line 30 to see if the motion of the missile has caused a collision.

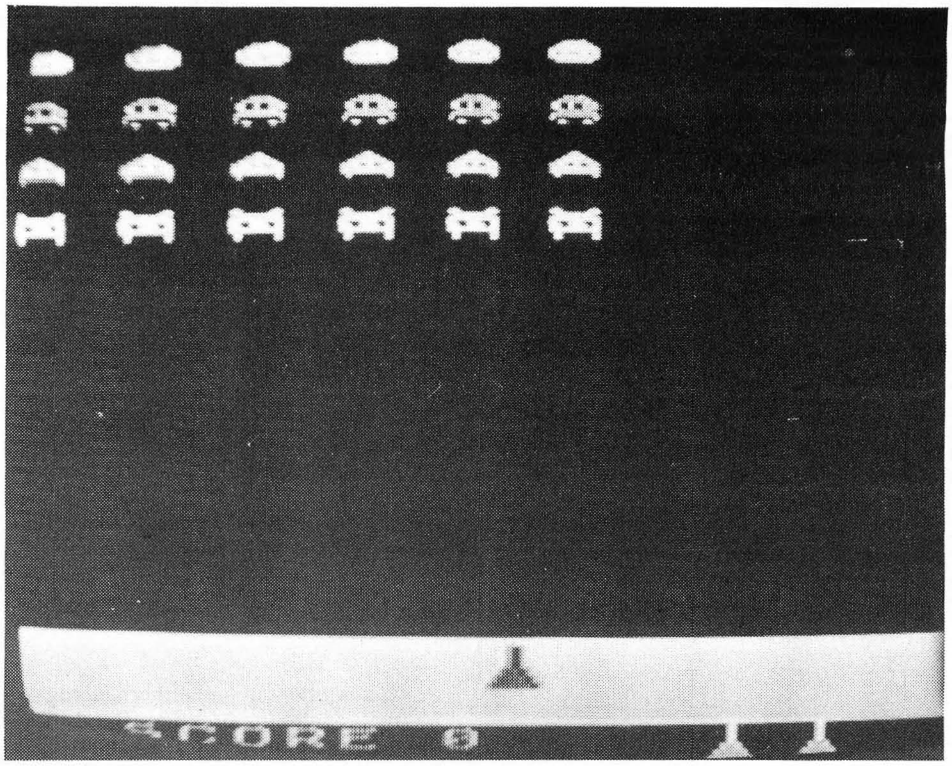
```
40 IF SF=1 THEN SVP=SVP-4:M$(SVP)=SHEL
L$:POKE 53278,1:IF SVP-4*K<10 THEN M$(
SVP,SVP+4)=BL$:SF=0
50 Y=PEEK(53248):IF Y>0 THEN GOSUB 210
```

Does the same kind of check as Line 40 does, only this time moves the bomb. Line 70 checks for a collision between the bomb and the laser base.

```
60 IF BF=1 THEN BVP=BVP+3:M$(BVP)=BOMB
$:IF BVP>102 THEN M$(BVP,BVP+5)=BL$:BF
=0:BVP=20
70 IF PEEK(53259)=1 THEN 250
```

Determines whether or not to drop a bomb (by setting the flag BF). As more screens are cleared, fewer iterations of the loop will pass before a bomb is dropped, and they will fall more frequently. The position of the bomb is determined by the position of the laser base, so they can fall out of nowhere, rather than being dropped by a particular invader.

```
80 IF BF=0 AND COUNT>ATTACK AND K<16 T
HEN BF=1:POKE 53255,HP+3:BVP=14+4*K:CO
UNT=0
```



Checks the joystick for right or left motion, and moves the player appropriately. No checking is done to see that the player is not moved off of the screen, because this would slow down the loop. Therefore, it is possible to crash the program by trying to move the player beyond the limits of the screen.

```
90 HP=HP+4*(STICK(0)=7)-4*(STICK(0)=11
):POKE 53248,HP
```

Checks the fire button, and if no missile is already in flight, it sets the missile flag and launches one. Line 110 merely adds one to the loop counter (COUNT), and returns to the main loop.

```
100 IF STRIG(0)=0 AND SF=0 THEN SF=1:H
PM=HP+2:POKE 53252,HPM:SVP=94:M$(SVP)=
SHELL$:POKE 77,0
110 COUNT=COUNT+1:RETURN
```

Lines 120-160: Recurring initialization routines.

```
120 POKE 656,1:POKE 657,2:? "GAME OVER
--PUSH BUTTON TO PLAY AGAIN";:SOUND 0,
0,0,0:CHANCES=0:SCREENS=0
130 POKE 656,1:POKE 657,2:? "
                                ";:POKE 704
,PEEK(710):IF STRIG(0)=1 THEN 120
```

Initializes the scoring routines and player positions for each screen. Also, variables are set up for the increasing game difficulty on succeeding screen. Lines 120-140 reinitialize the score total variable and "Game Over" message for the end of game.

```
140 POKE 656,3:POKE 657,2:? "SCORE   "
;:SCORE=0:GOSUB 240:POKE 705,38:POKE 7
06,38:ATTACK=80:TOUGH=0:TOUGHER=0
150 POKE 704,196:M$=BL$:SF=0:BF=0:POKE
 53278,1:FOR I=1 TO 3:FOR J=80 TO 150:
SOUND 0,J,10,6:NEXT J:NEXT I:COUNT=80
160 POKE 77,0:ATTACK=ATTACK-SCREENS*10
:IF SCREENS>1 THEN TOUGH=30:IF SCREENS
>3 THEN TOUGHER=60
```

Main program loop. Yes friends, the main loop that moves the invaders is only three lines long! All it takes to move them is to check the pointer to screen memory, and move it up one or back one position. The whole screen (except for the PM graphics) seems to move. The only reason it takes three whole lines is that the address pointer is in two bytes, and we have to check to see that the low byte value doesn't get under 0 or over 255. Also, the animation of the characters is accomplished by POKEing location 756 in line 180. This alternates the character sets being displayed. At several points during each iteration of the loop, the subroutine that moves the laser base, missiles and bombs is called.

```
170 FOR K=1 TO 20:D=-D:FOR I=0 TO 8:L=
PEEK(DL)-D-20*(I=8):H=PEEK(DH)-(L<0)+(
L>255):L=L+256*(L<0)-256*(L>255)
180 GOSUB 30:POKE DH,H:POKE DL,L:GOSUB
 30:POKE 756,RT+4+2*(PEEK(756)=RT+4):G
OSUB 30+TOUGH
190 SOUND 0,255,12,15:GOSUB 30+TOUGHER
:SOUND 0,0,0,0:NEXT I:NEXT K
200 GOSUB 420:M$=BL$:SF=0:BF=0:POKE 53
278,1:GOTO 120
```

Lines 210-270: Hit subroutines.

Contains the subroutine which erases invaders that have been hit, and scores the hit. Line 210 determines which character was hit.

```
210 SOUND 0,20,10,6:X=HPM+8*I*(-D)-64*
(D<0)-48:X=2*(INT(X/16)):Y=(Y=2)+3*(Y=
1)+5*(Y=8)+7*(Y=4)-1
220 SCORE=SCORE+10*(10-Y):HITS=HITS+1:
IF HITS=24 THEN POP :M$=BL$:SCREENS=SC
REENS+1:GOSUB 420:GOTO 150
230 POSITION X,Y:? #6;"  "
240 POKE 656,3:POKE 657,8:? SCORE;"
      ";:M$(SVP,SVP+2)=BL$:SF=0:POKE 5
3278,1:RETURN
```

Scores hits on the laser base. A spare base is erased, and if no spare bases are left, the game ends.

```
250 POKE 706-CHANCES,0:POKE 704,PEEK(7
10):M$=BL$:FOR K=70 TO 0 STEP -1:SOUND
 0,240,8,K/5:NEXT K:GOSUB 420
260 CHANCES=CHANCES+1:IF CHANCES=3 THE
N CHANCES=0:SCORE=0:GOTO 120
270 GOTO 150
```

Lines 280 to 460: Initialization routine. After initializing some important variables, the program jumps to line 280. The order of the program was carefully planned; the initialization routines which are only used once are placed at the back of the program, out of the way. The main loop goes in the middle, while frequently-called subroutines appear at the front. This layout adds much to the program's speed.

Checks the top-of-memory pointer and resets it, so that the computer will not use the top 8 pages. This space will be used for storing two new character sets, as well as the "moving" screen display. Line 290 makes the computer think that this area is the screen display, and zeroes out the memory there with a "clear screen" command (CHR$ 125). Next, the program calls a full-screen Graphics 3 display and sets the character-set pointer to the fourth page above the ramtop pointer (RT + 4).

```
280 RT=PEEK(106):IF RT/2=(INT(RT/2)) T
HEN RT=RT-8
290 POKE 89,RT:POKE 88,0:? CHR$(125):P
OKE 89,RT+3:? CHR$(125):POKE 106,RT:GR
APHICS 18:POKE 756,(RT+4)
295 ? #6;"% OUTER %":? #6:? #6;"    &
space &"
```

Prints the title, but since there is no character-set data stored at RT + 4 yet, nothing appears on the screen. Lines 310 to 340 fill in the two new character sets, mostly by reading the ROM set data, but changing the first four characters of each set with the data in lines 440-450. As this data is POKEd in, the title appears on screen.

```
300 ? #6:? #6;"          ' ATTACK '":? #
6:? #6:? #6:? #6;"   BY s. leemon"
310 FOR I=8 TO 39:READ X:POKE (RT+4)*2
56+I+32,X:SOUND 0,I,10,6:NEXT I
320 FOR I=40 TO 71:READ X:POKE (RT+6)*
256+I,X:SOUND 0,I,10,6:NEXT I
330 FOR I=72 TO 87:POKE (RT+6)*256+I-7.
2,0:SOUND 0,I,10,10:NEXT I
340 FOR I=98 TO 511:J=PEEK(57344+I):PO
KE (RT+6)*256+I,J:POKE (RT+4)*256+I,J:
SOUND 0,I,10,6:NEXT I
```

Stores the address of the pointer to display memory in DH and DL. Color registers are set in line 360.

```
350 FOR I=1 TO 600:SOUND 0,0,0,0:NEXT
I:GRAPHICS 1:POKE 559,0:DH=PEEK(560)+P
EEK(561)*256+5:DL=DH-1
360 POKE 89,RT+2:POKE 88,128:POKE 752,
1:? " ":POKE DH+25,6:POKE 708,52:POKE
709,132:POKE 710,8:POKE 711,84
```

Starts the initialization routine for ATARI® PM graphics. This program uses a most helpful trick: The space where Player/Missile data is stored is DIMensioned as a series of strings. Because PM$ must start on an even 1K boundary, we DIMension F$ to fill the space between D$ and the next highest 1K boundary. Now, whatever data is contained in the strings which are DIMensioned immediately after PM$ will appear on the screen as players and missiles. Line 380 zeroes out this memory, and line 390 reads the data for the shapes of the laser cannon, the shells, and the bombs into the appropriate strings. Lines 400-410 complete the Player/Missile initialization by setting the color registers, horizontal position registers, PM base address, and enabling double-line resolution PM graphics.

```
370 DIM BL$(128),D$(1),F$((INT(ADR(D$)
/1024)+1)*1024-ADR(D$)-1),PM$(384),M$(
128),P$(128),P1$(128),P2$(128)
380 DIM BOMB$(6),SHELL$(6):BL$=CHR$(0)
:BL$(128)=CHR$(0):BL$(2)=BL$:P$=BL$:M$
=BL$:SVP=88
390 FOR I=1 TO 6:READ K,Y,D:P$(97+I,97
+I)=CHR$(K):SHELL$(I,I)=CHR$(Y):BOMB$(
I,I)=CHR$(D):NEXT I
400 P1$=BL$:P2$=BL$:P1$(12)=P$:P2$=P1$
:POKE 53249,170:POKE 53250,156:POKE 70
4,196:POKE 707,76
410 BVP=10:POKE 53254,100:HP=128:POKE
53248,HP:POKE 54279,ADR(PM$)/256:POKE
623,1:POKE 53277,3
```

Sets screen memory to 2½ pages above ramtop. Line 430 prints the four rows of redefined characters that make up the invaders, and sends you back to the beginning.

```
420 POKE DH,PEEK(89):POKE DL,128:POKE
559,46:BF=1:HITS=0:D=-1:POKE 756,RT+4
430 ? #6;CHR$(125):FOR J=0 TO 3:POSITI
ON 0,J*2:FOR I=1 TO 6:? #6;CHR$(5+32*J
+J+64*(J>1));" ";:NEXT I:NEXT J:RETURN

440 DATA 60,126,255,165,255,126,0,0,60
,126,215,215,126,255,129,102,24,60,126
,235,255,255,65,0
445 DATA 195,126,255,153,221,255,65,65
,0,0,0,90,0,0,0,0,60,126,86,126,60,62,
34,54
450 DATA 24,60,126,215,255,255,130,0,1
95,66,126,255,153,187,255,65
460 DATA 24,1,0,24,1,0,24,0,0,60,0,0,1
26,0,192,255,0,192
```

Take-Apart: Outer Space Attack

by Sheldon Leemon

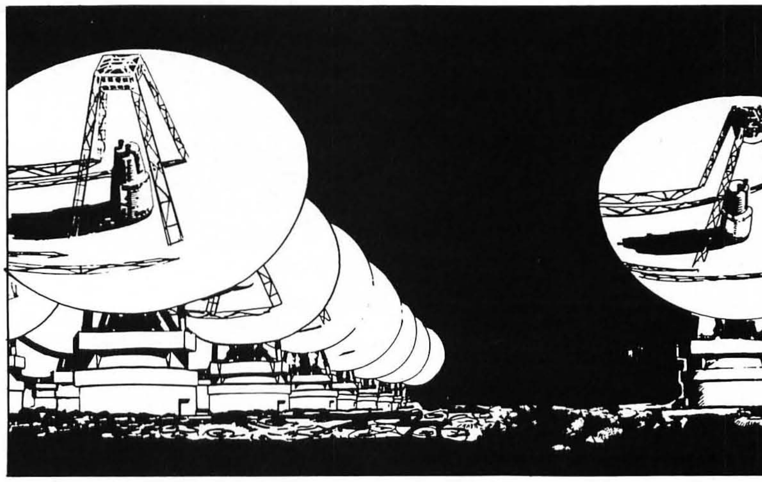
Outer Space Attack uses several of ATARI®'s exclusive graphics features. The first, character redefinition, has already been treated in an earlier Take-Apart article by Alan J. Zett which explained how to change the appearance of letters in order to create custom graphics characters. For *Outer Space Attack*, I used a character-generator program to create two sets of redefined characters, each portraying the Invader characters in different positions.

After designing the characters, the generator program then wrote the DATA statements in lines 440-450 of the program, which determine the Invaders' shapes. The POKE 756 command in line 180 switches back and forth between the sets, creating the illusion of animation.

The second graphics feature which the program takes advantage of is graphics indirection (modifying the display list). Other microcomputers use a fixed area of memory to hold the data being displayed on the screen, so that in order to move images on the screen, you usually have to move the screen data around within this area of memory.

The ATARI® computers, however, do not use a fixed area for display memory. Instead, two bytes of memory act as a pointer to display memory. By changing this pointer, it is possible to change the data being displayed without moving around any bytes in memory. You may think of this system as keeping a "window" on memory. Instead of moving the contents within the window, the frame is moved over the contents. In another sense, each separate line on the screen is a window.

Actually, two pointers are used to change the data being displayed. One tells the computer where to store data which is entered from the keyboard or PRINT statements (the "write pointer"), and the other tells the computer which area of memory to display (the "display pointer").



While these will usually have the same value so that the display shows what is written, it is quite possible to print words or graphics into one part of memory, while displaying another part, enabling you to flip instantly between "pages" without making the user watch the second screen being set up.

The write pointer is in memory locations 88 and 89. You can verify this by calculating the address pointed to with the statement WP = PEEK(88) + 256 *PEEK(89), and then using a POKE WP,33 command to make the letter "A" appear in the upper-left corner of the screen (screen data uses Internal Character Set values, shown on page 55 of the Reference Manual).

Outer Space Attack uses the write pointer to clear memory, by changing the pointer value to the memory address of the area used for character data, and printing CHR$(125) (Line 290). The "screen clear" character causes the computer to write 1024 zeroes to that area of memory, which it thinks contains the screen display.

In *Outer Space Attack*, the display pointer is changed frequently to give the Invaders the illusion of horizontal motion. This technique, known as coarse scrolling, is demonstrated in the following program:

```
10 DH = PEEK(560) + PEEK(561)
*256 + 5: DL = DH-1
```

First we find the two-byte pointer to display memory.

```
20 FOR I = 1 TO 40: POKE DL,
PEEK(DL) + 1: FOR J = 1 TO
50:NEXT J:NEXT I
```

Next we add one to the low-byte value of the pointer and then POKE the new value back in. This makes the whole display appear to move to the left. The loop which uses the J index was added to slow down the motion.

```
30 FOR I = 1 TO 40: POKE DL,
PEEK(DL) -1: FOR J = 1 TO 50:
NEXT J: NEXT I
```

This line repeats the motion, this time to the right.

These three lines show the basic logic used in the main program loop of *Outer Space Attack*. The only real difference is that there, provision was made to carry or borrow from the high byte when the low-byte value got above 255 or below 0 (see Line 170 of the program).

Although this explains the motion of the Invaders, why doesn't the base, the bombs, or the shells move at the same time? Because they are not being displayed by the regular screen graphics system, but by a completely separate system known as Player/Missile graphics. This system features up to four characters called players, each of which is eight screen dots wide, and four two-dot characters known as missiles. Each character is potentially as tall as the screen. The hardware maintains collision registers to record hits between the players, missiles, and graphics objects.

The topic of Player/Missile (PM) graphics is too broad to be dealt with easily in one short article. For those unfamiliar with this system, I would recommend that you read Chris Crawford's article in the November, 1981, issue of *Byte*, or the article published by Atari in their magazine *The ATARI® Connection*. But for those who are familiar with PM graphics, I will explain the trick used in *Outer Space Attack* to make them more manageable.

One problem with using PM graphics in BASIC programs is that vertical motion is usually produced by moving the player data through memory, POKEing one byte at a time. This produces slow and jerky motion. A better method exists. Normally, the user picks out an area in memory that is otherwise unused by the program in which to store data for PM graphics. But what if that area coincides with the data storage area of a string? Then, by manipulating the string data, you also will change the PM display data, at Machine Language speed.

The only obstacle to this approach is that PM data memory has to start on a 1K boundary (i.e., the first address must be an even multiple of 1024). The way that *Outer Space Attack* solves the problem is to dimension a string, F$, to take up space until the boundary is reached. In this way, the player data area is occupied by the string P$, and any time you change the value of P$, you change the player data.

The short example which follows demonstrates how easily PM graphics can be used in BASIC this way:

```
10 DIM D$(1), F$((INT
(ADR(D$)/1024)+1)
*1024- ADR(D$) -1),PM$(384),
M$(128), P$(128)
20 PM$=CHR$(0):
PM$(384)=CHR$(0):
PM$(2)=PM$: M$=PM$:
P$=M$
30 DIM BALL$(7): BALL$=P$:
FOR I=2 TO 6: READ A:
BALL$(I,I)=CHR$(A): NEXT I
40 DATA 24,24,24,255,24,24,24
50 POKE 704,14:POKE 54279,
ADR(PM$)/256: POKE 559,46:
POKE 53277,3: P$(60)=BALL$
60 FOR J=1 TO 5: FOR I=47 TO
200: POKE 53248,I: NEXT I: FOR
I=200 TO 47 STEP -1:POKE
53248,I: NEXT I: NEXT J:
P$=M$
70 POKE 53248,120
80 FOR J=1 TO 8: FOR I=16 TO
107: P$(I)=BALL$: NEXT I: FOR
I=107 TO 16 STEP -1:
P$(I)=BALL$: NEXT I: NEXT J:
P$=M$
```

Line 10 takes care of setting up the strings. PM$ represents a blank buffer before the start of the player memory, M$ is the area of missile memory, and P$ is the memory area used for Player 0. If the other three players are desired, these strings should be dimensioned right after P$. Line 20 fills PM$, M$, and P$ with all-zero data.

While touching on some of ATARI®'s unique features, this article in no way exhausts the possibilities. The hardware capabilities of these computers will let you go as far as you wish.

Line 30 then dimensions BALL$, and fills it with data that represents the shape of a ball. A zero is placed in front and behind the ball shape data, so that when the data is moved, the preceding and trailing zeroes will erase the remnants of the prior character.

Line 50 takes care of normal PM housekeeping. The first POKE sets the color of the ball to white. The second one makes the string area starting with PM$ the beginning of PM graphics data memory. The third POKE enables PM memory for double-line resolution graphics, and the fourth enables the player and missile display. Finally, the statement P$(60)=BALL$ sets the data at position 60 of player memory equal to the ball data, so that a ball will appear about midway down the screen.

The next two lines show how to move the ball. Line 70 demonstrates how horizontal position is changed by POKEing the horizontal position register (53248 decimal) with a value between 0 and 255. Notice that after this demonstration is finished, the player is cleared by assigning P$ the value of M$, which is all zeroes. Next, vertical motion is displayed, by sequentially changing the contents of P$. This produces smooth, fast motion without resorting to Machine Language routines.

As you can see, the techniques used in *Outer Space Attack*, though sophisticated, are not all that difficult to master. The small effort required to learn them will reward you with access to graphics power far beyond the reach of other micros. PM graphics alone offer effects which could not otherwise be achieved, by making four additional colors available for the screen display; allowing you to mix pictures and text on the same horizontal line; and giving you a method of creating smooth animation in BASIC, simply by defining several strings containing shape data, and alternately assigning to P$ the value of each. Other PM features such as selectable priority, which allows you to choose whether the player appears to go in front of or behind other screen graphics, and collision registers, which tell you when screen objects touch one another, lend themselves to sophisticated games.

While touching on some of ATARI®'s unique features, this article in no way exhausts the possibilities. The hardware capabilities of these computers will let you go as far as you wish.

I would like to thank the Michigan ATARI® Computer Enthusiasts, and especially Marcus Watts, who described to the group the PM techniques discussed here, for their help in developing *Outer Space Attack* and other programs.

K-razy Shoot-Out

A review by Sheldon Leemon

from K-BYTE. System requirements: ATARI® 400 or 800. Suggested retail price: ROM cartridge — $49.95.

With the introduction of *K-razy Shoot-Out*, a new firm called K-BYTE has made an impressive entry into the ATARI® computer software market. Not only has it produced a very professional arcade game, but it has also become the first company outside of Atari to offer this type of ROM cartridge software for the 400 or 800. Despite their relatively high prices, ROM-based games are very appealing to the consumer market, since they are reliable, durable, and easy to use, particularly for the minimum-configuration system owner who has relatively little computer experience. Since Atari has been actively cultivating consumers, and has been using such games to attract people who previously have been somewhat intimidated by computers, the appearance of second-source ROM packs is a significant event.

In *K-razy Shoot-Out*, your on-screen alter ego is a Space Commander whose mission is to rid Alien Control Sectors of the deadly Droids. In each of the maze-like Sectors, the Commander must destroy all Droids before the moving time bar disappears. If successful, he progresses to a new and even more hostile Sector. Bonus points are awarded if the job is finished before the time bar changes color.

Droids can be eliminated by using a laser weapon, or by tricking them into colliding with the charged barrier walls or each other. Collision with the barriers or Droids is also fatal to the Commander, and after the first level, the Droids return laser fire. A Commander who falls in battle is replaced by one of two backups. After the second one is gone, the game is over. The game will also end if the Commander has failed to leave the sector before the time bar disappears. In either case, the player's performance is evaluated and rank is awarded according to a complex formula based on the number of Droids destroyed, the Sector level reached, and the time required.

It is evident that the producers of this game have spent a lot of time working on the play values. While the first level can be mastered after a little practice, each new level introduces a quantum leap in difficulty, as the Droids get smarter, faster, and more numerous. The player's skills will gradually improve, but there is little danger that the game will become so easy that he will lose interest. Part of the reason for this is the random generation of Sector configurations, and random points of entry for replacement Droids (only three Droids are on the screen at a time). Because time counts for a good deal in the player evaluation formula and can add bonus points to your score, even when you have mastered the basic skills there is always room for improvement in speed. Taken as a whole, these elements give the game an extremely addictive quality, which is, after all, what arcade games are all about. I think it is fair to say that even after the initial excitement wears off, you will want to go back to this game in the future.

The player's skills will gradually improve, but there is little danger that the game will become so easy that he will lose interest.

All play functions are performed with the joystick and because of the pattern of action, the player must keep moving the stick in different directions. This helps to prevent his hand from turning into a useless claw after an extended period of play. The figure of the Commander changes color to indicate when there are only six more Droids left in a round. There is a recap at the end of each round, which gives the player a chance to relax. Also, the games can be restarted at any point by hitting the Start key. This saves a lot of frustration when you are trapped in an impossible Sector.

This is not to say that there isn't room for improvement. The graphics and sound effects are good, but not spectacular. Although the animation of the Commander and Droids is well done, the Sectors themselves are very plain. The ranking system shows no mercy. Even if you struggle along to an upper level, you may find that your rank has dropped back to Goon Class 5 (the lowest level) because you took too much time, or got too few points along the way. This continuing escalation of difficulty may be discouraging to those not hooked on arcade games. An option allowing you to skip the first couple of sectors would be a terrific addition. As things stand, you've got to play through all of the lower levels in order to get to the one which is currently driving you K-razy. Allowing multiple players would also be a very nice option to have. Unfortunately, there is only so much ROM available in the cartridge.

Despite its limitations, this program serves as a very nice introduction to K-BYTE. ATARI® computer owners should be quite encouraged by the direction that ROM-based software for their machine appears to be taking.

Random Files With ATARI® DOS I

by Odino Ciai and Luis Wuhl

This article may be a help for those people who own an ATARI® 800 with disk drives, but haven't bought or received ATARI® DOS II yet. (We are among them.) When using the ATARI® DOS I there are two problems with random files. First, if you know (a) where your file begins on the disk (sector and byte) and (b) the length in bytes of each one of your records, you should be able to access any record in the file with an easy calculation as shown in Table 1. However, this does not always work because you could POINT to a sector that is not within your file (if your file were not written in contiguous sectors), and the whole thing would get messed up. This might happen because DOS I writes on the disk using linked sectors.

Second, even if you overcome these problems, when you want to change some data in the file you can lose it. When you close a file (opened with OPEN#1,12,0,"D:FILE") just after writing in it, DOS I will sometimes rearrange the file's structure.

After some blood, sweat and tears, we have discovered a method (although an empirical one) to use random files with DOS I. The procedure and the rules are as follows:

1. Your file must be at least 128 bytes long (first trick); if not you must complete it with dummy characters.

2. You write the file for the first time as usual with OPEN#1,8,0,"D:FILE" and PRINT#1;Record but (second trick) before each PRINT, you must note the sector and byte where your record will be printed with NOTE#1,SECTOR,BYTE (SECTOR and BYTE are two variables) and store these values in a matrix (e.g., KEY(N,0)=SECTOR: KEY (N,1)=BYTE where N is the record number). This matrix will help you in future references to each record. After you have written all the records, you close the file and store the matrix KEY in the disk (e.g., under "D:FILE.KEY").

3. When you want to change a record you must open the file with OPEN#1,12,0,"D:FILE", point to the record you want with POINT#1,KEY(N,0),KEY(N,1), input the record for examination, change the record, point again with POINT#1,KEY(N,0),KEY(N,1), and print it with PRINT#1;Record, and so on for all your records that must be changed. But before closing the file, you MUST do a POINT to a sector and byte at least 128 bytes away from the last PRINTing (e.g., the beginning or the end of the file) and only after having done that you can close the file. Listings 1 and 2 are examples of the procedure.

We have been using random files with this method and we haven't had any trouble.

Note this other bug in DOS I: If you use a file longer than 256 sectors, DOS I does not reset the most significant byte in the directory and all the following files appear as being longer than they actually are. But don't worry: This is completely harmless.

CALCULATION OF POSITION FOR RANDOM FILES

Table 1

Suppose that having done a NOTE you have gotten SECTOR=3 and BYTE=0. Each sector of the disk has 125 usable bytes except the first of the file which has 126. Each of your records uses its length plus one for the EOL character.

The actual number of bytes of each sector is 128 but the DOS uses three of them for linkage purposes.

If your record length is 50 and you want to address record number 6 you must account the following:

BUSED=(6-1)*50 ;bytes used by the N-1 preceding records
SUSED=INT(BUSED/125)
BLINK=SUSED*3 ;bytes used by DOS
BTOTAL=BUSED+BLINK ;total of bytes used
STOTAL=INT(BTOTAL/128); total of sectors used
BYTE=BTOTAL-STOTAL*128; first byte of record number 6
SECTOR=STOTAL ;sector where record number 6 begins

Listing 1

```
1 REM This program writes the file.
10 DIM REG$(50),KEY(30,1):REM REG$ is
the string which holds each record.
20 OPEN #1,8,0,"D:FILE":REM This opens
the file for writing.
30 FOR A=1 TO 30:REM We will write 30
records.
40 FOR B=1 TO 50:REM Each record will
be 50 characters long.
50 REG$(B,B)=STR$(A):REM Each charac-
ter of the record is set to the record
number.
60 NEXT B
70 NOTE #1,SECTOR,BYTE:REM This sets
the sector and byte where the record
will be printed.
```

```
75 KEY(A,0)=SECTOR:KEY(A,1)=BYTE:REM
This stores those values for the
future use.

80 PRINT #1;REG$:REM Print the record
on the disk.

90 NEXT A
95 CLOSE #1:REM When the file is com-
plete, close the file.

100 OPEN #2,8,0,"D:FILE.KEY":REM Save
the matrix for future use.

110 FOR A=1 TO 30:PRINT #2;KEY(A,0):PR
INT #2;KEY(A,1)

120 NEXT A:CLOSE #2
```

Listing 2

```
1 REM Modification of a record.

10 REM Record number 5 will be changed
to a string of "+".

20 DIM REG$(50),KEY(30,1)

30 OPEN #1,4,0,"D:FILE.KEY"

40 FOR A=1 TO 30

50 INPUT #1,SECTOR,BYTE

60 KEY(A,0)=SECTOR:KEY(A,1)=BYTE

70 NEXT A

80 CLOSE #1

90 OPEN #2,12,0,"D:FILE"

100 POINT #2,KEY(5,0),KEY(5,1)

110 INPUT #2,REG$

120 PRINT REG$

130 REG$(1,1)="+":REG$(2)=REG$(1):REM
Sets each character of REG$ to "+".

140 PRINT REG$

150 POINT #2,KEY(5,0),KEY(5,1)
160 PRINT #2;REG$

170 POINT #2,KEY(30,0),KEY(30,1):REM
Must do this before closing.

180 CLOSE #2

200 REM Now you can examine the file
using the DOS command COPY (C) from
FILE to E:
```

GALACTIC CHASE™

The aliens have swept undefeated across the galaxy. You are an enterprising star ship captain—the final defender of space.

As the aliens attack, you launch a deadly barrage of missiles. Flankers swoop down on your position. Maneuvering to avoid the counterattack, you disintegrate their ships with your magnetic repellers.

As your skill improves, the attackers increase their speed. And as a last resort, the aliens use their invisible ray to slow the speed of your missile launcher.

GALACTIC CHASE provides Atari owners with the most challenging one or two person game in the galaxy.

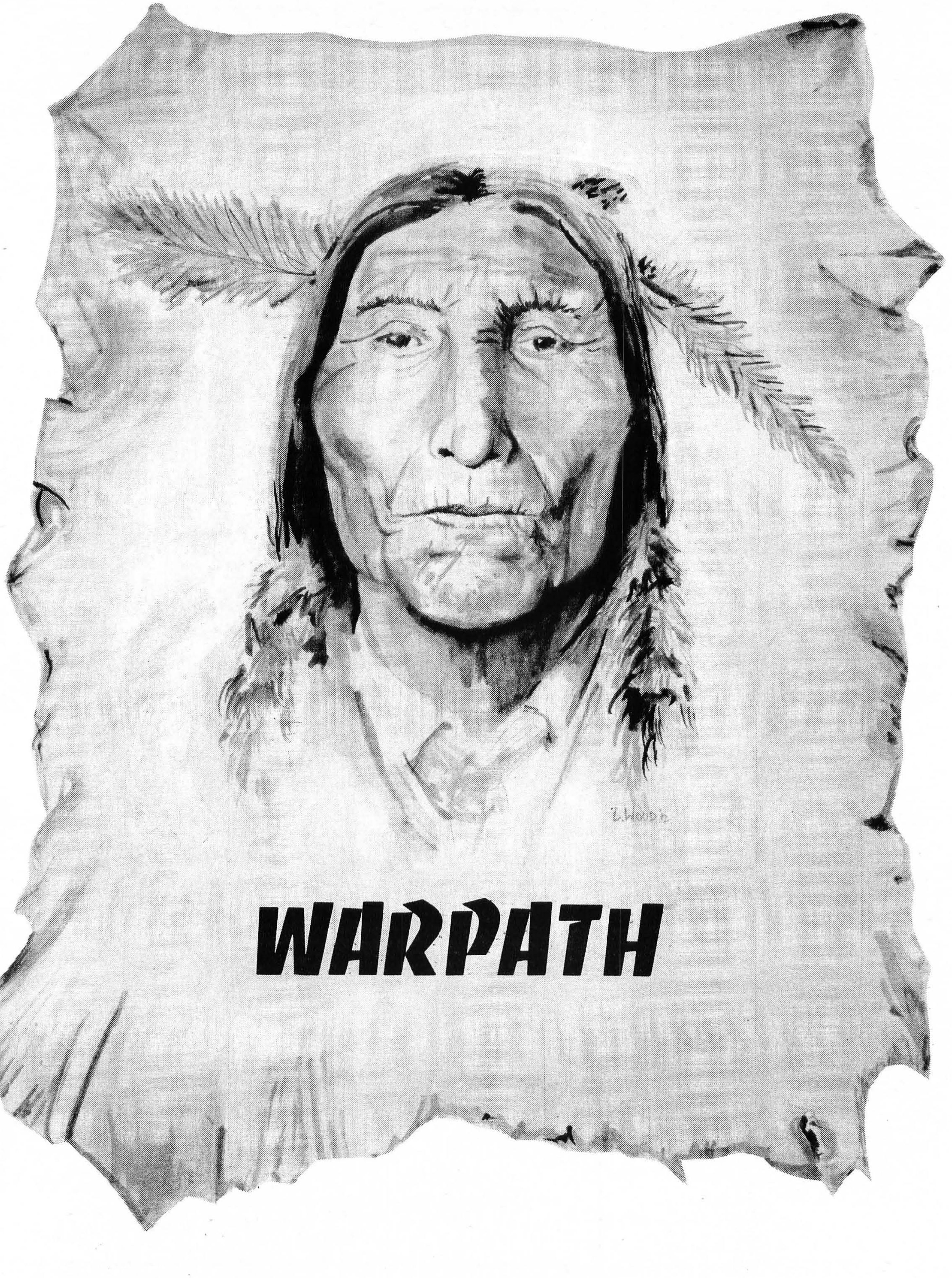
ATARI®

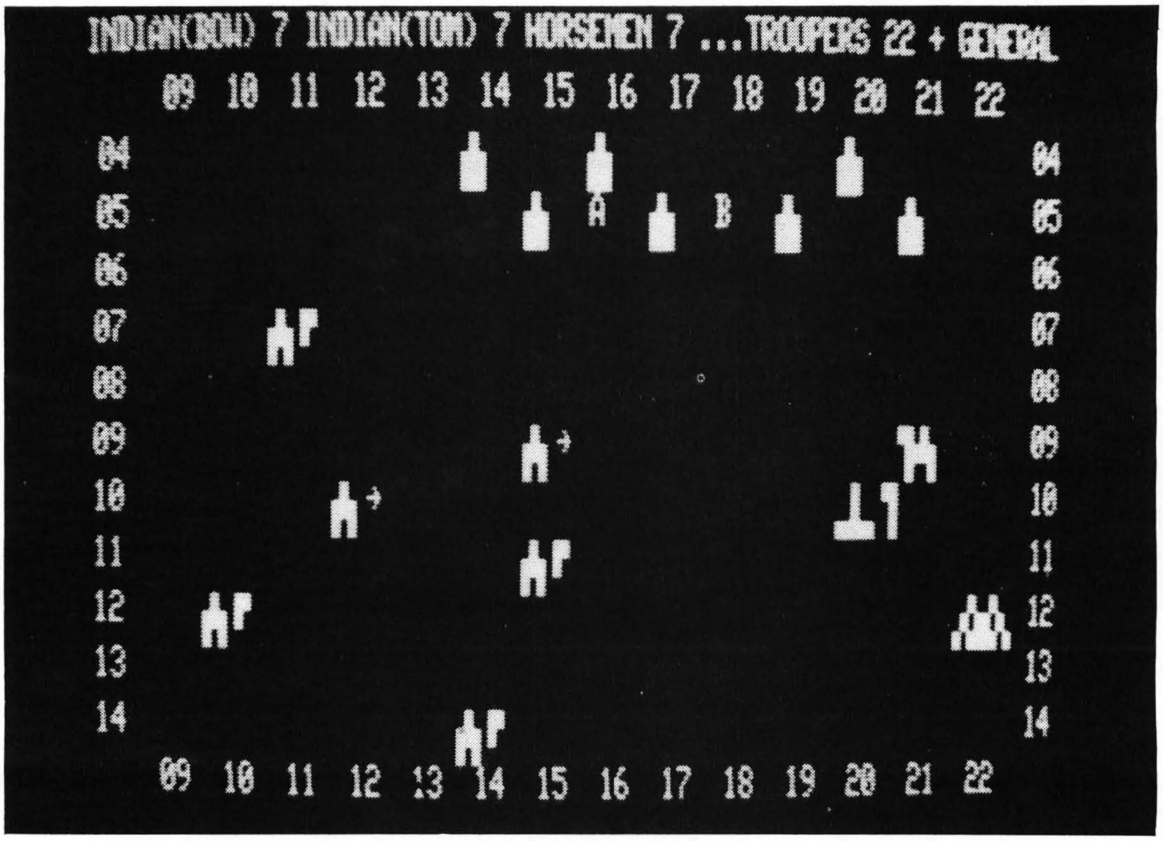
Atari 400/800 16k. Written in machine language. Requires joysticks.
Payment: Personal Checks—allow three weeks to clear.
American Express, Visa, & Master Charge—include all numbers on card. Please include phone number with all orders. 24.95 for cassette or 29.95 for disk plus 2.00 shipping. Michigan residents add 4%.

Check the dealer in your local galaxy. Dealer inquiries encouraged.
Galactic Chase © 1981 Stedek Software.

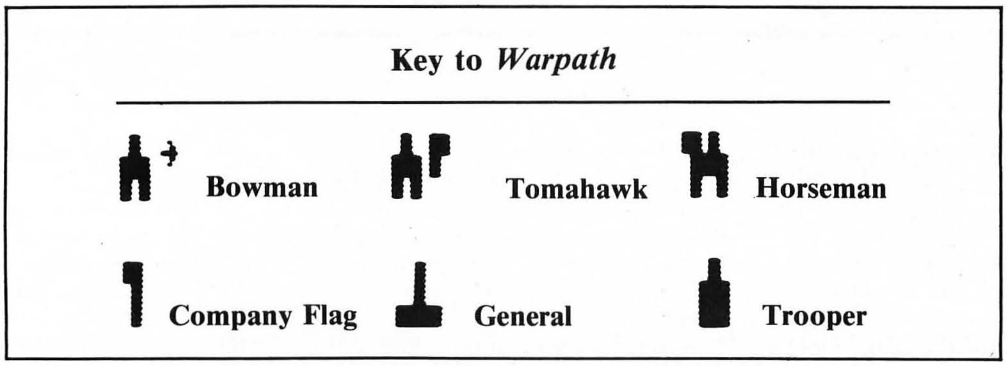
SPECTRUM
COMPUTERS

Dept S.
26618 Southfield
Lathrup Village, MI. 48076
(313) 559-5252


WARPATH



```
INDIAN(BOW) 7 INDIAN(TOM) 7 HORSEMEN 7 ...TROOPERS 22 + GENERAL
     09 10 11 12 13 14 15 16 17 18 19 20 21 22
04                                                04
05                      A     B                   05
06                                                06
07                                                07
08                                                08
09                                                09
10                                                10
11                                                11
12                                                12
13                                                13
14                                                14
     09 10 11 12 13 14 15 16 17 18 19 20 21 22
```

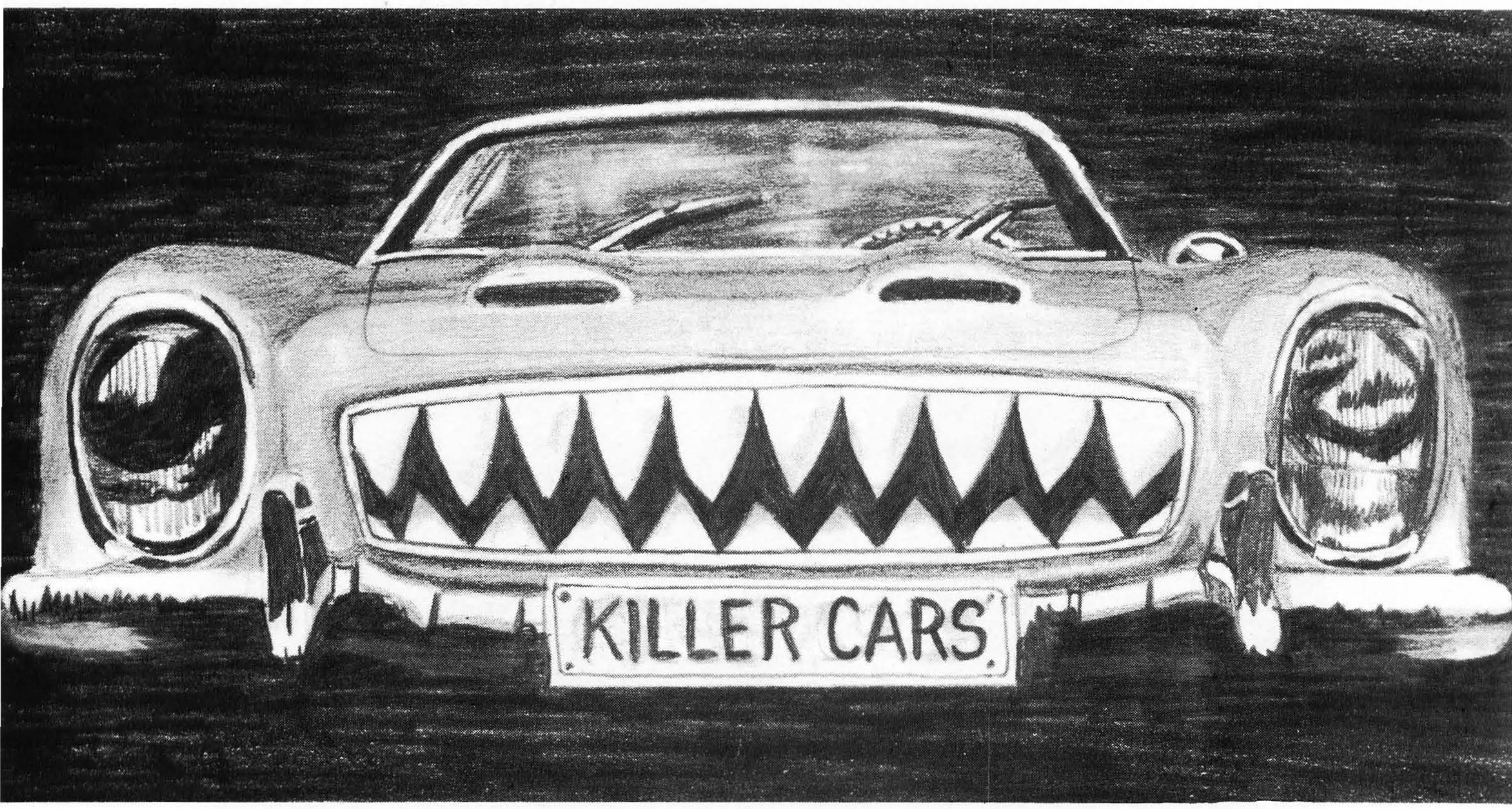


by Ron Potkin

***Warpath* is a wargame for a 32K TRS-80® Model I or III. It is included as a bonus program on this month's TRS-80® Disk Version.**

A party of Pawnee warriors, including bowmen, horsemen, and braves armed with tomahawks are creeping up on Fort Varptr, death and destruction their sole aim. The nine troopers in the garrison are becoming edgy as they wait for General Cursor and his reinforcements to arrive. The general, along with the company flag and some 14 troopers, must pick their way through a boulder-strewn plain, dispatching Indians with reckless abandon whenever the opportunity presents itself.

Warpath is a two-player wargame of strategy and skill. The Indians are trying to capture the fort, while the troopers are trying to prevent them from doing so, primarily through attrition. Should the reinforcements carry the flag into the garrison, the Indians will become disheartened and lose. On the other hand, should the Indians make it into the fort and slaughter the garrison, the troopers lose. Who will end up in the Happy Hunting Grounds? Boot up your disk and take to the WARPATH . . .



TRS-80®

by Richard Kipp

***Killer Cars* is a graphics game for a TRS-80® Model I or III with 16K RAM.**

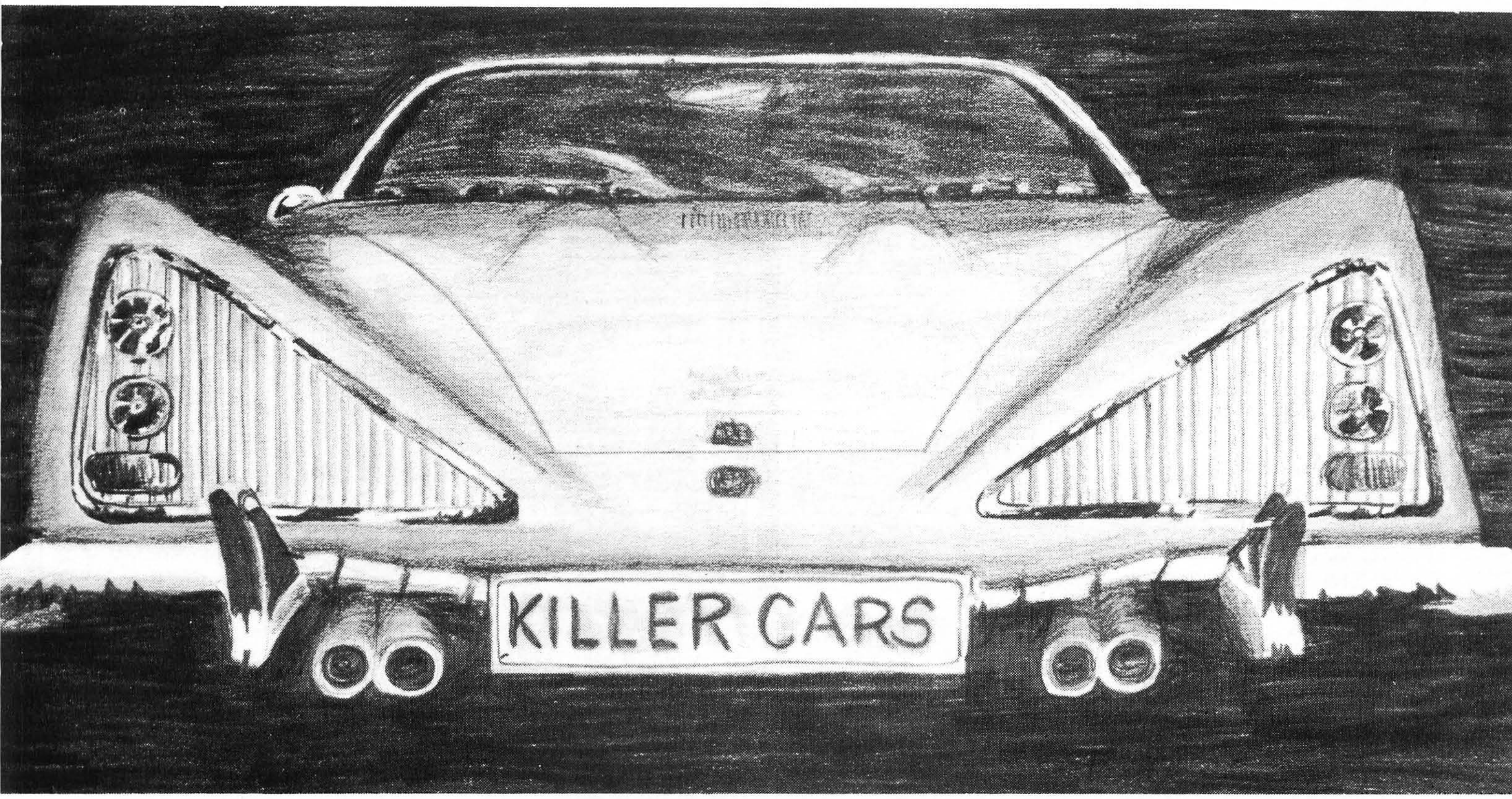
There is a popular video game in which two players are given identical armed vehicles and must destroy the opponent's tank while avoiding the opponent's weapons. *Killer Cars* takes this idea further by improving the simulation to give players more options and flexibility.

In this two-player game, you design your own vehicles, adding or subtracting parts to improve your chances of defeating your opponent. You each decide how large your engine will be, balancing power against efficient use of fuel. Should the vehicle have armor, and how much? Should oil be carried to create oil slicks? There are machine guns, cannons, and even lasers that can be mounted on the car. You can be sneaky and carry radio-controlled land mines, equip your vehicle with a ram, or hide behind powerful force shields. As you design your car, you should bear in mind that there is no free lunch. You will sacrifice mobility as you add hardware.

After designing your vehicle, you send it into battle against your opponent's. When a car's energy supply is exhausted, the game ends. For safety's sake, the cars are unmanned, and controlled by onboard computers. Each player programs his vehicle by sending it a radio message consisting of a string of ten commands. The commands that can be executed by the car's computer are as follows:

L: Turn car 45 degrees left.
R: Turn car 45 degrees right.
0 - 5: Move tank forward the specified number of meters, as allowed by the engine's power.
M: Fire machine guns. (All guns fire forward.)
C: Fire cannon.
B: Fire laser beams.
O: Drop oil slick.
S: Turn on force shield.
X: Turn off force shield.
D: Drop radio-controlled land mines.
E: Send the radio signal which causes all dropped mines to explode.
Any other symbol: Car does nothing.

When entering the string of commands, you should not press ENTER after each one; just type exactly ten characters, no more and no less, when prompted. If you do make a mistake, it can be corrected after you have typed in the ten commands and the computer asks "ALRIGHT?" (sic). The back-arrow key cannot be used to make corrections during command input.



TRS-80®

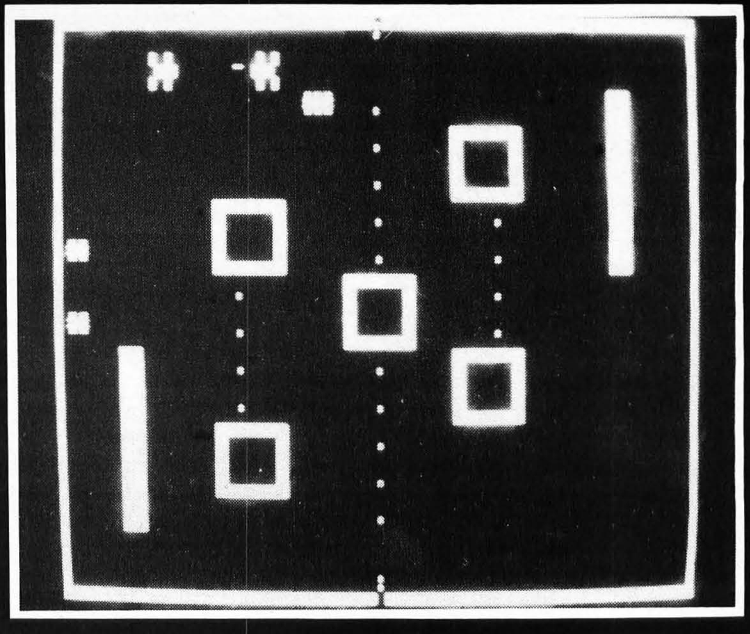
This program can be easily altered and enlarged. New commands can be added between lines 800 and 1000. Play time can be extended by setting TE(1) and TE(2) to 4000 during a break at the "HOW MANY HORSEPOWER TO DRIVES?" prompt. By changing the data in lines 1005 through 1180, a new playing field can be created. Ideas for other enhancements are as near as the closest "James Bond" spy movie.

Variables

B1, B2: Super graphics strings which hold the top and bottom half of the board.
BQ: String which holds Machine Language subroutine.
CC: The value of the command being executed.
CD, CL, CR, CS, CU, CV: Car graphics — facing down, left, right, diagonally, up, and reverse diagonally.
CM(x): Holds a string of the ten command letters entered during the command phase.
DP: Answer string for questions asked when building cars.
F, H, I, J, L, QK, R, TF, X, Y, Z!: General purpose counting variables.
GG: Random variable to determine whose actions are performed first.
K: The number of the player currently using the computer.
LD: Counter for time delay loop.
MD: Missile's damage.
MN(x): Number of land mines for each player.
MP: Missile's position.
MT: Missile's type.
OL(x): Number of oil barrels for each player.
PA(x): Player's armor thickness.
PD(x): Player's drive power.
PM(w,x): Positions of up to four mines for each player.
PP(x): Player's position (66-956).
QQ: The number of the player not currently using the computer.
RM(x): Player's ram (0 or 1).
SF(x): Player's force shield (0 or 1).
SM: Accumulator which calculates player's maximum speed at any one time.
SO(x): Value is 1 if player's vehicle is in oil, 0 if not.
SS(x): Status of player's shield (1 is on, 0 is off).
TD(x): Player's direction (1-8, as shown).

```
4   3   2
5   +   1
6   7   8
```

TE(x): Player's unused energy.
V: Sound routine dummy variable.
WC(x): Player's cannon (1 if he has one, 0 if not).
WL(x): Player's laser (0 or 1).
WM(x): Number of machine guns for each player (0-2).
XF: Count variable for the sound routine.
Z: Damage done by running through fence posts.



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         TRS-80 BASIC         $
$        "KILLER CARS"         $
$ AUTHOR: RICHARD L. KIPP II $
$     (C) 1982  SOFTSIDE       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization.

```
5 CLEAR600
7 DEFSTR A-E
11 B1=STRING$(160,170)
12 B2=STRING$(159,176)
15 DEFINTI-Z
17 CLS:GOSUB 9100
20 DIM TE(2),PA(2),PD(2),OL(2),WC(2),WM(2),WL(2),SF(2),RM(2),MN(
2),TD(2),PP(2),CM(2),SD(2),SS(2),PM(4,2)
```

Poke board graphics into B1 and B2.

```
55 RESTORE:FOR QQ=1TO27:READ XF:NEXTQQ
60 I=VARPTR(B1):J=PEEK(I+1)+256*PEEK(I+2)+65536*(PEEK(I+2)>=128)
70 FORK=JTOJ+158:READX:POKEK,X:NEXTK
85 RESTORE:FORX=1TO187:READI:NEXTX
90 I=VARPTR(B2):J=PEEK(I+1)+256*PEEK(I+2)+65536*(PEEK(I+2)>=128)
100 FORK=JTOJ+158
102 READ X
105 POKEK,X
106 NEXTK
```

Construct graphics car symbols.

```
160 CD=CHR$(185)+CHR$(182)
180 CU=CHR$(155)+CHR$(167)
200 CS=CHR$(172)+CHR$(158)+CHR$(141)
210 CR="":FORX=1TO3
220 READI:CR=CR+CHR$(I)
230 NEXTX
235 CV=CHR$(142)+CHR$(173)+CHR$(156)
240 CL="":FORX=1TO3
250 READI:CL=CL+CHR$(I)
260 NEXTX
```

Print title page.

```
300 CLS:PRINTB1;B2;:POKE 16383,186
302 FORX=1TO5:PRINT@148,"K I L L E R   C A R S ";:V=USR(5):FORY
=1TO40:V=USR(1266):NEXTY:PRINT@148,STRING$(23,128);:V=USR(25000)
:FORY=1TO40:V=USR(666):NEXTY:NEXTX
```

Construct your car.

```
303 TE(1)=RND(59)+1000:TE(2)=RND(59)+1000:LD=2000
304 V=USR(1)
305 FOR K=1TO2:CLS:PRINT"PLAYER"K,"TOTAL ENERGY ="TE(K)
310 INPUT"HOW MANY HORSEPOWER TO YOUR DRIVES(100-500)";F:IFF<100
ORF>500THEN310
320 PD(K)=INT(F):TE(K)=TE(K)-PD(K)*.8
330 INPUT"HOW MANY INCHES OF ARMOR (1-5)";PA(K):IFPA(K)>5THEN 33
0
340 INPUT"HOW MANY BARRELS OF OIL (0-3)";OL(K):IFOL(K)>3THEN340
350 INPUT"HOW MANY LAND MINES (0-4)";MN(K):IFMN(K)>4THEN350
355 INPUT"HOW MANY MACHINE GUNS (0-2)";WM(K):IFWM(K)>2THEN355
360 INPUT"DOES YOUR CAR HAVE A RAM";DP:IFLEFT$(DP,1)="Y"THENRM(K
)=1ELSERM(K)=0
370 INPUT"WILL YOU USE A FORCE FIELD";DP:IFLEFT$(DP,1)="Y"THENSF
(K)=1 ELSESF(K)=0
380 INPUT"WILL YOU USE A CANNON";DP:IFLEFT$(DP,1)="Y"THENWC(K)=1
ELSEWC(K)=0
400 INPUT"WILL YOU USE A LASER";DP:IFLEFT$(DP,1)="Y"THENWL(K)=1E
LSEWL(K)=0
```

Calculate car's weight and maximum speed.

```
410 PW(K)=0:PW(K)=PW(K)+PD(K)
420 PW(K)=PW(K)+OL(K)*150+PA(K)*105
425 PW(K)=PW(K)+WC(K)*200+WM(K)*75+WL(K)*125+SF(K)*150
430 PW(K)=PW(K)+RM(K)*200+MN(K)*50
435 CLS:PRINT"VEHICLE WEIGHT ="PW(K)
436 V=USR(12000)
437 SM=INT(PD(K)*9/PW(K)):IFSM>5THENSM=5
438 PRINT:PRINT"MAXIMUM SPEED:"SM;
439 GOSUB9300
440 NEXTK
```

Begin the game.

```
445 CLS:PRINTB1;B2;:POKE 16383,186
450 TD(1)=1:TD(2)=5:SC=1000:PP(1)=577:PP(2)=569
460 K=1:GOSUB5000:K=2:GOSUB5000
470 LD=999:GOSUB9300:PRINT@578,"1";:PRINT@570,"2";
```

Get commands from the players.

```
500 FORK=1TO2
505 PRINT@3,"PLAYER"K;:LD=777:GOSUB9300
510 PRINT@3,"COMMANDS: ";:DP=INKEY$:V=USR(5800)
511 PRINT@65,TE(K);:PRINT@12," ";
512 IFSF(K)=1ANDSS(K)=1THENSS(K)=0:UO=1:GOSUB5000:PRINT@12," ";
515 CM(K)="":DP=INKEY$:J=0
520 DP=INKEY$:IFDP=""THEN520
525 PRINTDP;:V=USR(506)
528 CM(K)=CM(K)+DP
530 J=J+1:IFJ<10 THEN 520
```

```
535 LD=1111:GOSUB9300:PRINT@3,STRING$(20,128);
536 V=USR(7000):PRINT@3," ";"ALRIGHT?";:DP=INKEY$
537 DP=INKEY$:IF DP="N" THEN505 ELSE IF DP=""THEN537
538 IFUO=1THENSS(K)=1:GOSUB5000
539 UO=0:PRINT@65,"     ";
540 V=USR(30198):V=USR(19876):V=USR(12345):NEXTK
545 PRINT@3,STRING$(20,131);:PRINT@65,"     ";
548 LD=2200:GOSUB9300
549 GG=RND(2)
```

Read, identify, and execute each player's instructions.

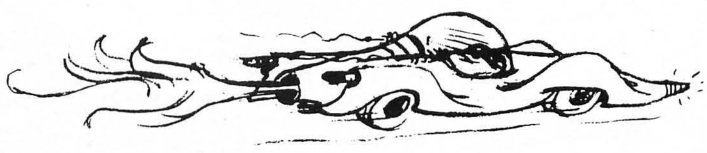
```
550 FORX=1TO10
560 IF GG>1 THEN 562 ELSE GOTO 563
562 FOR K=1TO2: GOTO565
563 FOR K=2TO1 STEP -1
565 CC=MID$(CM(K),X,1)
```

If player's vehicle is in oil, ignore steering commands.

```
567 IFSO(K)=1THEN590
```

Turn left.

```
604 FOR L=1TO I:FORXF=1TO150:NEXTXF
605 IFTD(K)=3ORTD(K)=7THENPRINT@PP(K),"  "; ELSE PRINT@PP(K),"
 ";
606 Y=15360+PP(K):IFTD(K)>1THEN 610 ELSE IF PEEK(Y+4)=32ANDPEEK(
Y+5)=32THENPP(K)=PP(K)+2
607 IFPEEK(Y+4)=35ORPEEK(Y+5)=35THENPRINT@PP(K)+3,"  ";:SO(K)=1:
PP(K)=PP(K)+2
608 IF PEEK(Y+4)=46 OR PEEK(Y+5)=46 THENPP(K)=PP(K)+2:Z=RND(50):
IFRM(K)=0AND SS(K)=0THENPD(K)=PD(K)-Z
609 GOTO699
610 IFTD(K)<>2THEN 620 ELSE IFPEEK(Y-63)=32 AND PEEK(Y-62)=32 AN
DPEEK(Y-61)=32THENPP(K)=PP(K)-63
611 IFPEEK(Y-63)=35ORPEEK(Y-62)=35ORPEEK(Y-61)=35THENPRINT@PP(K)
-63,"  ";:SO(K)=1:PP(K)=PP(K)-63
612 IF PEEK(Y-63)=46ORPEEK(Y-62)=46ORPEEK(Y-61)=46THEN PP(K)=PP(
K)-63:Z=RND(100):IFRM(K)=0ANDSS(K)=0THENPD(K)=PD(K)-Z
613 GOTO699
620 IF TD(K)<>3THEN626 ELSE IF PEEK(Y-64)=32ANDPEEK(Y-63)=32THEN
 PP(K)=PP(K)-64
622 IF PEEK(Y-64)=46 ORPEEK(Y-63)=46THENPP(K)=PP(K)-64:Z=RND(130
):IFRM(K)=0ANDSS(K)=0THENPD(K)=PD(K)-Z
623 IF PEEK(Y-64)=35ORPEEK(Y-63)=35THENPRINT@PP(K)-64,"  ";:PP(K
```



```
570 IFCC<>"L"THEN580 ELSETD(K)=TD(K)+1:IFTD(K)=9THENTD(K)=1
572 IFTD(K)<>4ANDTD(K)<>8ANDPEEK(15363+PP(K))<128 THENPRINT@PP(K
),"  ";
575 TE(K)=TE(K)-1:GOSUB5000:FOR XF=1TO55:V=USR(888):NEXT XF:GOTO
1000
```

Turn right.

```
580 IFCC<>"R"THEN589 ELSE TD(K)=TD(K)-1:IFTD(K)<1THENTD(K)=8
582 IFTD(K)<>2ANDTD(K)<>6ANDPEEK(15363+PP(K))<129 THENPRINT@PP(K
),"  ";
585 TE(K)=TE(K)-1:GOSUB5000:FOR XF=1TO55:V=USR(888):NEXT XF:GOTO
1000
589 IFTF>0THENGOSUB3400
```

Move forward.

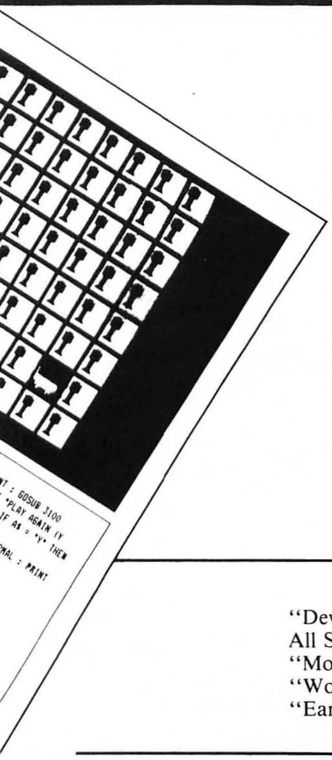
```
590 IFCC>CHR$(48)ANDCC<CHR$(54)THENIF TD(K)=3 OR TD(K)=7 THENPRI
NT@PP(K),"  "; ELSE PRINT@PP(K),"  "; ELSE GOTO 700
592 SM=INT(PD(K)*9/PW(K)):IFSM<1THEN700 ELSE IF SM>5THENSM=5
600 I=VAL(CC)
601 SO(K)=0
602 IF I>SM THEN I=SM
```

Movement routines for each of eight possible directions.

```
)=PP(K)-64:SO(K)=1
624 GOTO699
626 IFTD(K)<>4THEN632 ELSE IF PEEK(Y-67)=32ANDPEEK(Y-66)=32ANDPE
EK(Y-65)=32THENPP(K)=PP(K)-65
628 IFPEEK(Y-67)=46ORPEEK(Y-66)=46ORPEEK(Y-65)=46THENPP(K)=PP(K)
-65:Z=RND(100):IFRM(K)=0ANDSS(K)=0THENPD(K)=PD(K)-Z
630 IFPEEK(Y-67)=35ORPEEK(Y-66)=35ORPEEK(Y-65)=35THENPP(K)=PP(K)
-65:SO=1
631 GOTO699
632 IFTD(K)<>5THEN640 ELSE IFPEEK(Y-2)=32ANDPEEK(Y-1)=32THENPP(K
)=PP(K)-2
634 IFPEEK(Y-2)=46ORPEEK(Y-1)=46THENPP(K)=PP(K)-2:Z=RND(50):IFRM
(K)=0ANDSS(K)=0THENPD(K)=PD(K)-Z
636 IFPEEK(Y-2)=35ORPEEK(Y-1)=35THENPP(K)=PP(K)-2:SO=1
638 GOTO699
640 IFTD(K)<>6THEN648 ELSE IFPEEK(Y+61)=32ANDPEEK(Y+62)=32ANDPEE
K(Y+63)=32THENPP(K)=PP(K)+61
642 IFPEEK(Y+61)=46ORPEEK(Y+62)=46ORPEEK(Y+63)=46 THENPP(K)=PP(K
)+61:Z=RND(100):IFRM=0ANDSS(K)=0THENPD(K)=PD(K)-Z
644 IFPEEK(Y+61)=35ORPEEK(Y+62)=35ORPEEK(Y+63)=35 THENPP(K)=PP(K
)+61:SO=1
646 GOTO699
648 IFTD(K)<>7THEN656 ELSE IFPEEK(Y+64)=32ANDPEEK(Y+65)=32 THENP
P(K)=PP(K)+64
650 IFPEEK(Y+64)=46ORPEEK(Y+65)=46 THENPP(K)=PP(K)+64:Z=RND(130)
:IFRM(K)=0ANDSS(K)=0THENPD(K)=PD(K)-Z
```

SoftSide
January 1982 Vol. 5, No. 4 THREE DOLLARS
Off and running with GAMBLER
December 1981 Vol. 5, No. 3 THREE DOLLARS
TITAN
November 1981 Vol. 5, No. 2 THREE DOLLARS
Special Issue: Music in the Micro
Flight of the Bumblebee
Your BASIC Software Magazine • NUMBER ELEVEN
APPLE CAPTURE by William J. Ryan
TRS-80™ Version

Back Issues from SoftSide™



VOLUMES FROM THE PAST

If you like what this issue of **SoftSide** has to offer, you should see what's waiting for you in **SoftSide** back issues! You may feel that you've missed out on many of our programs that appeared before you became a subscriber. It's not too late to do something about it. You won't have to miss a thing because we still have back issues available to complete your **SoftSide** library! But, order now as some of our more popular issues are already out of stock and others are dwindling quickly.

Listed below are all of our past issues with their FEATURE programs and the systems they're for. For a more complete index of all the programs and articles offered in each of the back issues of **SoftSide** please refer to the May, 1981, issue. Each issue costs $3.50 for the magazine only. These issues are also available with the programs on cassette for $9.95 or on disk for $14.95 (except DV issues).

The enhanced Disk Versions (DV) contain an extra program for each system. The TRS-80® DV began with the September, 1981, issue. The Apple DV began October, 1981, and the ATARI® DV started in November, 1981. Each enhanced DV costs $19.95.

October 1980
"Developing Data Base II"
All Systems
"Moonlanding" — Apple
"World Series" — ATARI®
"Earth-Port II" — TRS-80®

November 1980
"Developing Data Base III" —
All Systems
"Collision" — Apple
"Trench" — ATARI®
"Kriegspiel" — TRS-80®

December 1980
"Developing Data Base IV"
All Systems
"Baseball" — Apple
"Speedello" — ATARI®
"Kidnapped" — TRS-80®

January 1981
"Developing Data Base V"
All Systems
"Convoy" — Apple
and TRS-80®
"Angle Cannon" — ATARI®
"Ship Destroyer" — TRS-80®

February 1981
"Developing Data Base VI"–
All Systems
"Miner" — All Systems
"Mini-Golf" — ATARI®
and TRS-80®
"Long Distance" — TRS-80®

March 1981
"Developing Data Base
VII" — All Systems
"Strategy Strike" — Apple
and TRS-80®
"Flags" — ATARI®
"Volcano" — TRS-80®

April 1981
"Battle At Sea" — Apple
"Convoy" — ATARI®
"Dominoes" — TRS-80®

May 1981
"Galaxia" — Apple
"Dodge" — ATARI®
"Orienteering At Jacque's
Coulee" — TRS-80®

June 1981
"Old Glory" — All Systems
"Word-Search Puzzle
Generator" — All Systems
"Anallist" — TRS-80®

July 1981
"Chemistry Drill" —
All Systems
"Kidnapped" — Apple
and ATARI®
"Magic Paper
Calculator" — TRS-80®

October 1981
"Leyte" — All Systems
"Developing Data Base"—Apple
"Character Generator" —
ATARI®
"Envyrn™" — TRS-80®
Enhanced Disk Versions
"Super Dairy Farming" — Apple
"Gameplay" — TRS-80®

November 1981
"Flight of the Bumblebee" —
All Systems
"Music Machine" — Apple
"Music Programmer"—ATARI®
"Music Editor" — TRS-80®
Enhanced Disk Versions
"National Anthems" — Apple
"Volleyball" — ATARI®
"Mean Checkers Machine" —
TRS-80®

December 1981
"Titan" — All Systems
"Aircraft Commander" — Apple
"Developing Data
Base" — ATARI®
"Electronics
Assistant" — TRS-80®
Enhanced Disk Versions
"Bobsledding" — Apple
"Survive" — ATARI®
"Konane" — TRS-80®

January 1982
"Gambler" — All Systems
"Microtext 1.1" — All Systems
"Apple Capture" — Apple
"Piazza Hotel" — ATARI®
"TRS-Man" — TRS-80®
Enhanced Disk Versions
"Nuclear Submarine Adventure"
Apple, TRS-80®
"Death Star" — ATARI®

February 1982
"Space Rescue" — All Systems
"Rubicube" — Apple
"Defense" — ATARI®
"Maze Sweep" — TRS-80®
Enhanced Disk Versions
"Andorra" — Apple
"Kismet II" — ATARI®
"Help Package" — TRS-80®

Use the card provided in this issue to order or send a list of the back issues you'd like with payment of $3.50 per magazine to:

SoftSide Publications
515 Abbott Drive
Broomall, PA 19008

To order the magazine/media combinations, use the card provided or send $9.95 per cassette and magazine, $14.95 per disk and magazine, or $19.95 per DV and magazine to:

SoftSide Publications
6 South Street
Milford, NH 03055

```
652 IFPEEK(Y+64)=35ORPEEK(Y+65)=35THENPP(K)=PP(K)+64:SO(K)=1
654 GOTO699
656 IF PEEK(Y+65)=32ANDPEEK(Y+66)=32ANDPEEK(Y+67)=32ANDPEEK(Y+68
)=32THENPP(K)=PP(K)+65
666 IFPEEK(Y+65)=46ORPEEK(Y+67)=46ORPEEK(Y+66)=46THENPP(K)=PP(K)
+65:Z=RND(100):IFRM(K)=0ANDSS(K)=0THENPD(K)=PD(K)-Z
676 IFPEEK(Y+65)=35ORPEEK(Y+66)=35ORPEEK(Y+67)=35THENPP(K)=PP(K)
+65:SO(K)=1
699 TE(K)=TE(K)-1:GOSUB5000:GOSUB9400:NEXTL:GOSUB1800:GOTO1000
```

Turn on shield.

```
700 IFCC<>"S"THEN710 ELSEIFSF(K)=1THENSS(K)=1:V=USR(30000):V=USR
(22000):GOTO1000
```

Turn off shield.

```
710 IFCC<>"X"THEN720 ELSESS(K)=0:V=USR(22000):V=USR(30000)
720 IFSF(K)=1ANDSS(K)=1THENTE(K)=TE(K)-1
```

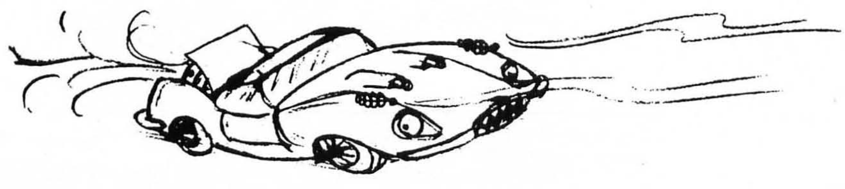
Drop oil.

Drop land mine.

```
760 IFCC<>"D"THEN 770
762 IFMN(K)<1THEN1000 ELSE MN(K)=MN(K)-1
764 PM(MN(K),K)=PP(K):PW(K)=PW(K)-50
768 GOTO1000
```

Detonate all dropped bombs.

```
770 IFCC<>"E"THEN800
772 IFMN(K)<0 ORMN(K)>3 THEN1000 ELSEIFK=1THENR=2 ELSER=1
774 FORQK=0TO4
776 IFPM(QK,K)=0THEN794
777 V=USR(9000):V=USR(260+RND(5000))
778 IFABS(PM(QK,K)-PP(R))<3 ORABS(PM(QK,K)-PP(R)+64)<2 ORABS(PM(
QK,K)-PP(R)-64)<2 THEN 780 ELSE GOTO790
780 IFSS(R)=1THEN790
782 MD=RND(150):IFMD<30 PD(R)=PD(R)-MD
784 IFPA(R)=0THENTE(R)=TE(R)-MD*1.2 ELSETE(R)=TE(R)-MD/PA(R)
790 PRINT@PM(QK,K),"MINE";:LD=30:GOSUB9300:PRINT@PM(QK,K)," ** "
;:GOSUB9300:PRINT@PM(QK,K),CHR$(RND(63)+128)CHR$(191)CHR$(RND(63
)+128);
```



```
730 IFCC<>"O"THEN736
732 IFOL(K)<1THEN1000
733 V=USR(28000)
734 OL(K)=OL(K)-1:TF=PP(K):PW(K)=PW(K)-145
735 GOTO1000
```

Fire cannon.

```
736 IFCC<>"C"THEN742
737 MT=1:V=USR(20295)
738 MD=20+RND(80):IFWC(K)=1 GOSUB3500
740 GOTO1000
```

Fire machine gun.

```
742 IFCC<>"M"THEN750
743 FOR XF=1TO9:V=USR(1400):FOR X8=1TO10:NEXT X8: NEXTXF
744 MT=2:MD=WM(K)*RND(35):GOSUB3500
746 GOTO1000
```

Fire laser.

```
750 IFCC<>"B"THEN760
752 MT=3:MD=WL(K)*(30+RND(94)):TE(K)=TE(K)-5
754 IFSS(K)=1THEN1000
756 IFWL(K)=1THENV=USR(1):GOSUB3500
758 GOTO1000
```

```
792 GOSUB9300:PRINT@PM(QK,K),"    ";
794 PM(QK,K)=0:NEXTQK
800 GOTO1000
```

Loop back for next player's instructions.

```
1000 IF TE(1)<1ORTE(2)<1 THEN 1200 ELSE GOSUB5000:SC=SC-1:NEXTK:
NEXTX:GOTO500
```

Data for sound routine.

```
1002 DATA 205,127,10,77,68,62,1,105,211,255,45,32,253,60
1003 DATA 105,211,255,45,32,253,13,16,238,175,211,255,201
```

Data for screen graphics (top half).

```
1005 DATA151,131,131,131,131,131,131,131,131,131,131,131,131,131
,131,131,131,131,131,131,131,131,131,131,131,131,131,131,131,131
,131,58,131,131,131,131,131,131,131,131,131,131,131,131,131,131,
131,131,131,131,131,131,131,131,131,131,131,131
1010 DATA131,131,131,131,131,171
1020 DATA149,222,32,223,170
1030 DATA149,222,46,215,191,191;198,170
1040 DATA149,222,46,199,191,131,131,131,131,131,191,201,191,191,
198,170
1050 DATA149,222,46,199,191,176,176,176,176,176,191,201,191,191,
198,170
```

```
1060 DATA149,206,191,131,131,131,131,131,191,201,46,203,46,203,1
91,191,198,170
1070 DATA149,206,191,176,176,176,176,176,191,201,46,203,46,203,1
91,191,198,170
1080 DATA149,208,46,202,191,131,131,131,131,131,191,200,46,211,1
70
```

Data for screen graphics (bottom half).

```
1100 DATA149,208,46,202,191,176,176,176,176,176,191,200,46,211,1
70
1110 DATA149,197,191,191,201,46,205,46,199,191,131,131,131,131,1
31,191,209,170
1120 DATA149,197,191,191,201,46,205,46,199,191,176,176,176,176,1
76,191,209,170
1130 DATA149,197,191,191,199,191,131,131,131,131,131,191,201,46,
223,170
1140 DATA149,197,191,191,199,191,176,176,176,176,176,191,201,46,
223,170
1150 DATA149,197,191,191,215,46,223,170
1160 DATA149,222,32,223,170
1170 DATA181,176,176,176,176,176,176,176,176,176,176,176,176,176
```

```
K)-RND(20)/PA(K):GOSUB9000
1860 IFSF(QQ)=1 ANDSS(QQ)=1 THEN1870 ELSEIFPA(QQ)=0 THENTE(QQ)=T
E(QQ)-RND(75) ELSETE(QQ)=TE(QQ)-RND(60)/PA(QQ)
1863 GOSUB9000
1865 FORZ!=1TO40:NEXTZ!
1870 PRINT@PP(QQ),CHR$(141);CHR$(191);CHR$(141);:RETURN
```

Place oil spots in unoccupied spaces.

```
3400 PRINT@TF,"#";:Y=15360+TF
3410 IFPEEK(Y+1)=32THENPOKEY+1,35:IFPEEK(Y+2)=32THENPOKEY+2,35:I
FPEEK(Y+3)=32THENPOKEY+3,35
3420 IFPEEK(Y-1)=32THENPOKEY-1,35:IFPEEK(Y-2)=32THENPOKEY-2,35:I
FPEEK(Y-3)=32THENPOKEY-3,35
3430 IFPEEK(Y-64)=32THENPOKEY-64,35
3440 IFPEEK(Y-65)=32THENPOKEY-65,35
3450 IFPEEK(Y-63)=32THENPOKEY-63,35
3460 IFPEEK(Y+63)=32THENPOKEY+63,35
3470 IFPEEK(Y+64)=32THENPOKEY+64,35
3480 IFPEEK(Y+65)=32THENPOKEY+65,35
3485 TF=0
3490 RETURN
```

```
,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176
,176,58,176,176,176,176,176,176,176,176,176,176,176,176,176,176,
176,176,176,176,176,176,176,176,176,176,176,176
1180 DATA176,176,176,176,176
1190 DATA 153,174,132,174,140,145
```

End-of-game routine.

```
1200 LD=444:GOSUB9300:IFTE(1)<1THEN K=1 ELSE K=2
1210 PRINT@18,"PLAYER"K"IS OUT OF ENERGY!!!";:IFK=1THENY=2 ELSEY
=1
1215 H=K:K=Y:GOSUB5000:K=H:FORX=1TO3:GOSUB9000:NEXTX
1220 FORZ=1TO85:GOSUB5000:V=USR(257+RND(300)):PRINT@PP(K),"   ";
:NEXTZ:PRINT@91,"THE WINNER!";:PRINT@158,SC;:V=USR(1):LD=4444:GO
SUB9300
1230 PRINT@513,"PLAY AGAIN";:V=USR(21111):INPUTDP:IFLEFT$(DP,1)=
"Y"THENRUN ELSEEND
```

Ram the other vehicle.

```
1800 Z=0:IFK=1THENQQ=2 ELSEQQ=1
1820 IF ABS(PP(K)+64-PP(QQ))<4 OR ABS(PP(K)-PP(QQ))<=4 OR ABS(PP
(K)-PP(QQ)-64)<4 THEN Z=1
1840 IFZ=1THENPRINT@PP(QQ),"<#>"; ELSE RETURN
1845 IFRM(K)=1 OR SF(K)=1ANDSS(K)=1THEN1860
1850 IF PA(K)=0THEN TE(K)=TE(K)-RND(20):GOSUB9000  ELSETE(K)=TE(
```

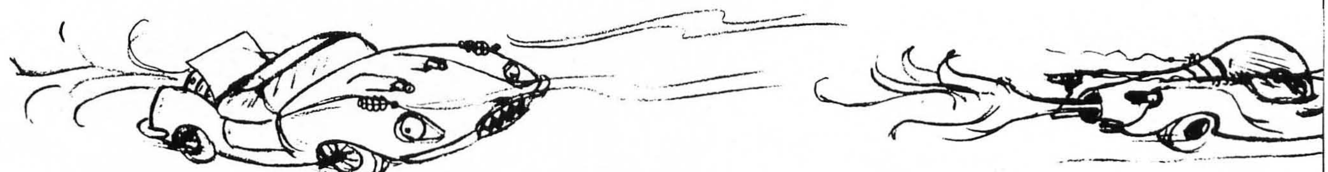
Missile weapon routine. First determine projectile and direction.

```
3500 MP=PP(K):Y=PP(K)+15360:IFK=1THENR=2 ELSE R=1
3501 IFTD(K)=1THENY=Y+3:MP=MP+3
3502 IFSS(K)=1ANDMT<3THENRETURN
3503 IFTD(K)=1ORTD(K)=5THENL=40:D1="-":GOTO3508
3504 IFTD(K)=3ORTD(K)=7THENL=14:D1="!":GOTO3508
3506 L=9:IFTD(K)=2ORTD(K)=6THEND1="/" ELSE D1="V"
3508 H=0:L=L-1:J=MP:IFL<1THENPRINT@MP," ";:RETURN
3509 Z!=MP/64:Z!=Z!-INT(Z!):IFZ!<.016 OR Z!>.96THENL=0:GOTO3508
```

Send missile in one of eight directions.

```
3510 ON TD(K) GOTO3520,3530,3540,3550,3560,3570,3580,3590
3520 MP=MP+1:Y=Y+1:IFPEEK(Y)=32THEN 3640
3521 IFPEEK(Y)=35THENH=1:GOTO3640
3522 IFABS(MP-PP(R))<3THEN3650
3524 GOTO3670
3530 Y=Y-62:MP=MP-62:IFABS(MP-PP(R))<3THEN3650
3532 IFPEEK(Y)=32THEN3640
3533 IFPEEK(Y)=35THENH=1:GOTO3640
3534 GOTO3670
3540 PRINT@MP," ";:Y=Y-64:MP=MP-64:IFMP<0 RETURN
3542 IF ABS(MP-PP(R))<3THEN3650
3544 IFPEEK(Y)=32THEN3640
```

```
3545 IFPEEK(Y)=35THENH=1:GOTO3640
3550 Y=Y-66:MP=MP-66:IFMP<0 RETURN
3552 IFABS(MP-PP(R))<3THEN3650
3554 IFPEEK(Y)=32THEN3640
3555 IFPEEK(Y)=35THENH=1:GOTO3640
3556 GOTO3670
3560 Y=Y-1:MP=MP-1:IFMP<0THENL=0:GOTO3640
3562 IFABS(MP-PP(R))<3THEN3650
3564 IFPEEK(Y)=32THEN3640
3565 IFPEEK(Y)=35THENH=1:GOTO3640
3566 GOTO3670
3570 Y=Y+62:MP=MP+62:IFABS(MP-PP(R))<3THEN3650
3572 IFPEEK(Y)=32THEN3640
3573 IFPEEK(Y)=35THENH=1:GOTO3640
3574 GOTO3670
3580 Y=Y+64:MP=MP+64:IFABS(MP-PP(R))<3THEN3650
3582 IFPEEK(Y)=32THEN3640
3583 IFPEEK(Y)=35THENH=1:GOTO3640
3584 GOTO3670
3590 Y=Y+66:MP=MP+66:IFABS(MP-PP(R))<3THEN3650
3592 IFPEEK(Y)=32THEN3640
3593 IFPEEK(Y)=35THENH=1:GOTO3640
```



```
3594 GOTO3670
```

Missile hitting border, houses, or other vehicle.

```
3640 PRINT@J," ";:IFH=1THENPRINT@J,"*";:H=0
3641 IFMP>1022ORMP<65THEN L=0:GOTO3508
3642 IFMT=2THENPRINT@MP,".";:GOTO3508
3643 IFMT=1 PRINT@MP,CHR$(140);:GOTO3508
3644 PRINT@MP,D1;:GOTO3508
3650 PRINT@J," ";:PRINT@MP," ";:IF SF(R)=1ANDSS(R)=1THENMD=MD/4
3652 IFPA(R)>0THEN MD=MD/PA(R)
3653 GOSUB9000
3654 TE(R)=TE(R)-MD:PRINT@PP(R)+1,"*";:FORZ!=1TO27:NEXTZ!:IFTD(R
)=3 ORTD(R)=7THEN PRINT@PP(R),CHR$(153);CHR$(166); ELSE PRINT@PP
(R),CHR$(153)"x"CHR$(166);
3656 PD(R)=PD(R)-RND(MD/10):H=K:K=R:GOSUB5000:K=H:RETURN
3670 PRINT@J," ";:IFMP<65THENPRINT@MP,CHR$(131);:PRINT@MP+64," "
;:RETURN
3671 IFMP>959THENPRINT@MP,CHR$(176);:RETURN
3672 IFMD>100THENPRINT@MP," ";
3675 RETURN
```

Print the car in its present position.

```
5000 IF PP(K)<0 ORPP(K)>1020THENPP(K)=75:TE(K)=TE(K)-16
5002 IFTD(K)=3ORTD(K)=7 THENPRINT@PP(K)," "; ELSE PRINT@PP(K),"
  ";
5003 IFSS(K)=1ANDSF(K)=1THENPRINT@PP(K),CHR$(191);CHR$(191);:RET
URN
5005 ON TD(K) GOTO5010,5080,5030,5020,5050,5080,5070,5020
5010 PRINT@PP(K),CR;:RETURN
5020 PRINT@PP(K),CV;:RETURN
5030 PRINT@PP(K),CU;:RETURN
5050 PRINT@PP(K),CL;:RETURN
5070 PRINT@PP(K),CD;:RETURN
5080 PRINT@PP(K),CS;:RETURN
```

Sound for damage to opponent's car.

```
9000 FORXF=300TO900 STEP 30:V=USR(XF):NEXT XF:RETURN
```

Initialize Machine Language sound routine.

```
9100 BQ="123456789012345678901234567"+CHR$(47)
9110 I = VARPTR(BQ): J = PEEK(I+1)+256*PEEK(I+2)+65536*(PEEK(I+2
)>=128)
9120 FOR KL=JTOJ+26: READ LK: POKE KL,LK: NEXT KL
9130 IF PEEK(16396)=201 POKE 16526,PEEK(I+1):POKE 16527,PEEK(I+2
):GOTO9150
9140 CMD "T": DEFUSR0=PEEK(I+1)+256*PEEK(I+2)+65536*(PEEK(I+2)>=
128):POKE 14308,0
9150 RETURN
```

Time delay subroutine.

```
9300 FOR QQ=1TOLD:NEXTQQ:RETURN
```

Sounds for movement of different vehicles.

```
9400 FOR XF=1TO15
9410 IF PA(K)=5 THEN V=USR(512):GOTO9460
9420 IF PA(K)=4 THEN V=USR(500):GOTO9460
9430 IF PA(K)=3 THEN V=USR(450):GOTO9460
9440 IF PA(K)=2 THEN V=USR(425):GOTO9460
9450 IF PA(K)=1 THEN V=USR(350):GOTO9460
9455 IFPA(K)=0 THEN V=USR(305)
9460 NEXT XF: RETURN
```

STAKE YOUR CLAIM

by Rik Pierce

Stake Your Claim **is a strategy game for a TRS-80® Model I or III computer with 16K RAM. (Special note: Media subscribers will receive an enhanced version containing packed strings and additional sound routines.)**

Have you ever thought what it would have been like to take part in the gold rush of 1849? With rumors of vast amounts of gold waiting to be uncovered, many thousands of people left the relative safety of their homes for the chance of becoming rich.

During this era staking a claim on the piece of land where you intended to dig was essential. And you always had to be on the lookout for claim jumpers.

As you climb down from your covered wagon, before you lies an area of land five units across by five units wide. Because of claim jumping, strict laws have been set down pertaining to the claiming of land:

1. Neither player may, as a first choice, claim the center square of land. This leads to too many disputes concerning who gets the first choice.
2. Neither player may claim a square that directly faces a square (either horizontally or vertically) that is claimed by another. You may stake a claim on any square next to your own squares, or on an empty square, except those squares mentioned above.

If these rules are kept, peaceful settlement of land may be achieved. How well you do depends upon your skill which can be tested against one other person or the computer.

A round ends when there are no squares left that can be legally taken. Scoring is based on the difference in the number of squares staked by each player. For example, if one player scores 5 and the other 8, the first player receives 0 points for the round while the second player scores 3 points. The game continues until either player has reached a score of 7.

Plan your moves carefully and you may win the game. But above all, STAKE YOUR CLAIM!

Variables

AG(n): Number of games won by player n.
AZ, AZ$: Miscellaneous (listed version only).
AZ$(1)-AZ$(5): Sound strings for winning music (listed version).
B(n): Square owned by player n.
D1: Count of players' squares.
D2: Count of squares where play is illegal.
D$: PRINT string to move cursor down one line and backspace three.
F1: Playing board string.
GL(1)-GL(25): Character graphics for introduction.
N: Player number.
PO: Points gained during a round.
Q: PRINT @ location on board.
Q1-Q3: Used during initialization. Also used to balance or skew the computer's choice. These may be changed to alter computer play.
SK(n): A flag. If equal to five, player n cannot play.
S$: String used to hold sound data for the sound routine.
T$(n): Graphics token for player n.
T$(3): Graphics token for TRS-80® when choosing a move.
TR(n): A flag. If equal to 99, then player n is TRS-80®.
TIM: Used in timing loops.
V: Memory location for start of video; set to 15360.
WE$: In media version, this is the packed string for winning music.

TRS-80®

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$   TRS-80 Mod I/III BASIC     $
$     "STAKE YOUR CLAIM"       $
$     AUTHOR: Rik Pierce       $
$     (c) 1982  SoftSide       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Reserve high memory according to amount available for Machine Language subroutines.

```
1 CLEAR:POKE16561,111:POKE16562,127-64*(PEEK(16562)>127)
```

Determine if program is running on a Model I or III.

```
2 CLEAR1000:TRS=1:IFPEEK(293)=73THENTRS=3
```

Test for lower case.

```
3 POKE15360,97:IFTRS=3ORPEEK(15360)<>97THEN10
```

POKE in lower-case driver

```
4 READA$:IFA$<>"HHH"THEN4ELSEMT=PEEK(16562)*256+255
5 MT=MT+65536*(MT>32767):FORI=MT-40TOMT:READB:POKEI,B:NEXT
7 POKE16414,215:POKE16415,PEEK(16562):RESTORE
```

POKE in sound routine.

```
10 CLS
20 PRINT@386,CHR$(23)"Plug in aux cable for sound"
25 READA$:IFA$<>"MMM"THEN25ELSEMT=PEEK(16562)*256+255
30 MT=MT+65536*(MT>32767):FORI=MT-131TOMT-68:READB:POKEI,B:NEXT
```

Adjust sound routine to look at S$ for notes to be played.

```
40 S$="":V=VARPTR(S$)
50 Q3=INT(V/256):Q4=V-Q3*256:Q0=MT-131
70 Q3=Q3-256*(Q3<0):POKEQ0+1,Q4:POKEQ0+3,Q3
```

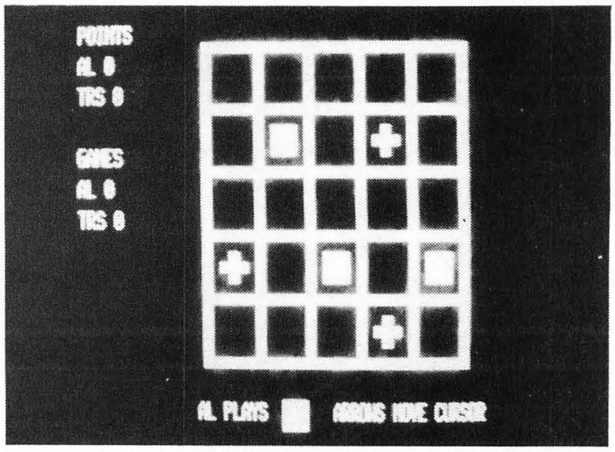
Set up USR call for tape or disk.

```
80 IFPEEK(16396)=201THENAZ=Q0-65536*(Q0<0):POKE16527,INT(AZ/256)
:POKE16526,AZ-PEEK(16527)*256:GOTO100
90 DEFUSR=Q0:CMD"T":POKE14308,0
```

Define graphics strings and dimension arrays.

```
100 DEFSTRF-G
110 DIMP(5,5),P1(5,5),I(14),J(14),GL(25),PP(25),Y(4,4),Z(14)
120 F1=CHR$(191)+"     ":F1=F1+F1+F1+F1+F1+CHR$(191)
130 D$=CHR$(26)+STRING$(3,24)
140 T$(1)=STRING$(3,188)+D$+STRING$(3,179):C(2)=1
150 T$(2)=CHR$(176)+CHR$(188)+CHR$(176)+D$+CHR$(176)+CHR$(179)+C
HR$(176):C(1)=2
160 T$(3)="   "+D$+CHR$(24)+STRING$(5,176)
170 G(1)=STRING$(3,191):G(2)=CHR$(140)+CHR$(191)+CHR$(140)
180 G=CHR$(176)
```

Calculate and fill arrays with PRINT @ positions.



```
190 V=15360
200 W=28:Y=5
210 P=(Y+1)/3*64+(W/2+2)
220 X=0:FORI=0TO4:FORJ=0TO4
230 P(I,J)=P+J*6+I*128
240 IFCN=1ORCN=3ORCN=2ANDX=9THEN260
250 X=X+1:PP(X)=P(I,J):I(X)=I:J(X)=J
260 NEXTJ
270 CN=CN+1
280 NEXTI:CN=0
```

Set up sound for 64-character mode.

```
290 CLS:POKEMT-105,1:POKEMT-91,2
```

Set up for main game display.

```
300 GOSUB1510:GOSUB3510
310 FORX=1TO14
320 R=RND(14):IFZ(R)=99THEN320ELSEZ(R)=99
325 Y=Y+1:IFY>2THENY=1
327 PRINT@P(I(R),J(R)),T$(Y);:FORTIM=1TO300:NEXT
330 PRINT@P(I(R),J(R)),GL(R);
340 FORTIM=1TO70:NEXT
350 NEXT X
360 FORTIM=1TO500:NEXT
370 PRINT@855,"by Rik Pierce"
```

Offer instructions.

```
380 S$=PS$:U=USR(0):FORTIM=1TO200:NEXT:PRINT@855,"Instructions?"
390 S$=S$(1):U=USR(0)
400 I$=INKEY$:IFI$=""THEN400
410 IFI$="Y"THENGOSUB2500
```

Input and adjust player names; display player pieces.

```
430 FORX=1TO2:PRINT@855,CHR$(30)"Player "X;:INPUTN$(X)
440 S$="5"+CHR$(40-5*X):U=USR(0)
460 N$(X)=LEFT$(N$(X),10)
470 IFLEN(N$(X))<2 THEN 550
480 IFLEFT$(N$(X),3)="TRS"THENTR(X)=99:GOTO530
```

```
530 PRINT@910,N$(X)", you will play the "G(X)"  piece   "
540 FORTIM=1TO200:NEXT
550 NEXTX
560 N=0
620 FORTIM=1TO1500:NEXT
800 '
```

Main program loop. Redraw display, check for end of game, and get player moves.

```
810 GOSUB1510
820 PRINT@64,"POINTS";:PRINT@128,N$(1);S(1);:PRINT@192,N$(2);S(2
);
830 PRINT@320,"GAMES";:PRINT@384,N$(1);AG(1);:PRINT@448,N$(2);AG
(2);
840 Q=P(2,2)+1
850 II=2:JJ=2:I=2:J=2
1000 '
1010 N=N+1:IFN=3THENN=1
1040 IFTR(N)=99THENPRINT@845,CHR$(30);N$(N)" is thinking";:GOSUB
2000
1050 D1=0:D2=0:FORI=0TO4:FORJ=0TO4
1060 IFP1(I,J)=N THEN D1=D1+1
1070 IFP1(I,J)>2 THEN D2=D2+1
1080 NEXTJ,I
1090 IFD2=25THEN1560
1100 IFSK(N)=5 THEN 1000
```

Check if player is unable to move.

```
1110 IFD1+D2<25 THEN 1140 ELSE SK(N)=5:PRINT@430,"No more moves
for";:PRINT@494,N$(N)" ";
1120 IFLEN(N$(N))>7 THEN PRINT@558,;
1130 PRINT"this turn";:GOSUB3500:GOTO1000
1140 PRINT@845,CHR$(31)N$(N)" plays  "G(N)"   ";:PC=POS(X)
1142 S$=S$(N):U=USR(0)
1145 IFTR(N)=99THENI=I1:J=J1:GOTO1700
```

Move square-select cursor.

```
1150 IFPEEK(14400)>0THEN1150
1160 PRINT"arrows move cursor"
1170 PK=PEEK(14400)
1180 IFPK=8ANDI>0THEN II=I-1:JJ=J
1190 IFPK=16ANDI<4THEN II=I+1:JJ=J
1200 IFPK=32ANDJ>0THEN II=I:JJ=J-1
1210 IFPK=64ANDJ<4THEN II=I:JJ=J+1
1220 IFPK=1THEN1700
1230 A=PEEK(V+P(II,JJ)+1)
1240 F=CHR$(A)
1250 A1=176
1260 IFA=140THENA1=188
1270 IFA=188THENA1=140
1280 I=II:J=JJ:Q=P(I,J)+1
1290 G=CHR$(A1)
1310 PRINT@Q,G;:PRINT@Q,F;:FORTIM=1TO20:NEXT
1320 IFPK>0 THEN PRINT@832+PC,"ENTER stakes the claim"
1330 GOTO1170
1500 '
```

Display game board.

```
1510 PRINT@128,;:FORZ=1TO10:PRINTTAB(W/2)F1:NEXT
1520 PRINT@P(0,0)-66,STRING$(31,176);
1530 FORX=0TO4:FORY=0TO4:P1(X,Y)=0:PRINT@P(X,Y),T$(3);:NEXTY,X
1540 FORZ=1TO2:B(Z)=0:SK(Z)=0:NEXT
1550 RETURN
```

End-of-round update.

```
1560 PRINT@845,"This round is over. ";
1570 PO=ABS(B(1)-B(2))
1580 IFPO=0 THEN PRINT"No gains this round":GOTO1635
1590 IFB(1)>B(2)THENN=1ELSEN=2
1600 S(N)=S(N)+PO :PRINTCHR$(30)N$(N)" gains"PO"point";:IFPO>1TH
ENPRINT"s"
```

Check for legal end of game.

```
1610 IFS(N)>6THENPRINT@909,"    "N$(N)" WINS the game":S$=PS$:U=U
SR(0):FORTIM=1TO600:NEXT ELSE1635
1620 IFTR(C(N))=99THENPRINT"You have done the IMPOSSIBLE! You ha
ve BEATEN THE COMPUTER!";
1630 GOSUB2700:GOTO1670
```

Cycle for next round.

```
1635 IFTR(1)=99ANDTR(2)=99THENFORTIM=1TO1500:NEXT:GOTO800
1640 S$=S3$:U=USR(0):PRINT@974,"press SPACEBAR for next round";
1650 I$=INKEY$:IFI$<>" "THEN1650
1660 GOTO800
```

Update player totals for games won.

```
1670 AG(N)=AG(N)+1:S(1)=0:S(2)=0
1672 IFTR(1)=99ANDTR(2)=99THEN1680
```

Prompt for next game.

```
1675 PRINT@832,CHR$(31):PRINT@914,"Another game?"
1676 I$=INKEY$
1677 I$=INKEY$:IFI$=""THEN1677
1678 IFI$="N"THENCLS:CLEAR50:END
1680 GOTO800
1700 '
```

Check move legality.

```
1710 IFI=2ANDJ=2ANDB(1)=0ANDB(2)=0THEN1890
1720 IFP1(I,J)=NORP1(I,J)>2THEN1900
1730 P1(I,J)=3
1740 IFI=0THEN1770
1750 IFP1(I-1,J)=0THENP1(I-1,J)=C(N)
1760 IFP1(I-1,J)=NTHENP1(I-1,J)=3
1770 IFP1(I+1,J)=0THENP1(I+1,J)=C(N)
1780 IFP1(I+1,J)=NTHENP1(I+1,J)=3
1790 IFP1(I,J+1)=0THENP1(I,J+1)=C(N)
1800 IFP1(I,J+1)=NTHENP1(I,J+1)=3
1810 IFJ=0 THEN 1840
1820 IFP1(I,J-1)=0THENP1(I,J-1)=C(N)
1830 IFP1(I,J-1)=NTHENP1(I,J-1)=3
1840 IFTR(N)=99 THENFORX=1TO20:PRINT@P(I,J)+1,CHR$(176);:PRINT@P
(I,J),"  ";:NEXT
1850 PRINT@P(I,J),T$(N);
1870 B(N)=B(N)+1
1880 GOTO1000
```

Illegal-move routine.

```
1890 PRINT@910,"First play may NOT be in the center";:GOTO1910
1900 PRINT@906,"CANNOT place piece NEXT to opponent's piece";:
1910 S$="!Z!"+CHR$(1):FORX=1TO5:U=USR(0):NEXT:FORTIM=1TO750:NEXT
:PRINT@896,CHR$(31);
1920 GOTO1170
2000 '
```

Calculate computer's move.

```
2010 Q1=1.5 ' IF ADJACENT PLACE IS UP FOR GRABS (=0)
2020 Q2=2   ' IF ADJACENT PLACE CAN BE SPOILED (=N)
2030 Q3=-2  ' IF ADJACENT PLAY CANNOT BE OWNED (=3)
2035 IFRND(2)=1THENQ4=Q1:Q1=Q2:Q2=Q4
2040 IFSK(N)=5THEN2400
2050 IFB(1)=0ANDB(2)=0THENI1=RND(3):J1=RND(3):IFI1=2ANDJ1=2THEN2
050ELSE2400
2060 FORI=0TO4:FORJ=0TO4
2070 I2=RND(5)-1:J2=RND(5)-1:AF=PEEK(V+P(I2,J2)+1)
2080 IFAF=32THENFF=CHR$(176)
2090 IFAF=188THENFF=CHR$(140)
2100 IFAF=140THENFF=CHR$(188)
2110 PRINT@P(I2,J2)+1,FF;
2120 IFP1(I,J)=NORP1(I,J)=3THEN2340
2130 'IFP1(I,J)=C(N)THENSC=SC+3
2140 IFI=0THEN2180
2150 IFP1(I-1,J)=0THENSC=SC+Q1
2160 IFP1(I-1,J)=N THEN SC=SC+Q2
2170 IFP1(I-1,J)=3THENSC=SC+Q3
2180 IFI=4THEN2220
2190 IFP1(I+1,J)=0THENSC=SC+Q1
2200 IFP1(I+1,J)=NTHENSC=SC+Q2
2210 IFP1(I+1,J)=3THENSC=SC+Q3
2220 IFJ=4THEN2260
2230 IFP1(I,J+1)=0THENSC=SC+Q1
2240 IFP1(I,J+1)=NTHENSC=SC+Q2
2250 IFP1(I,J+1)=3THENSC=SC+Q3
2260 IFJ=0 THEN 2300
2270 IFP1(I,J-1)=0THENSC=SC+Q1
2280 IFP1(I,J-1)=NTHENSC=SC+Q2
2290 IFP1(I,J-1)=3THENSC=SC+Q3
2300 '
2310 IFSC=S1THENCN=CN+1:I(CN)=I:J(CN)=J
2320 IFSC>S1THENS1=SC:I1=I:J1=J:CN=0
2330 SC=0
2340 PRINT@P(I2,J2)+1,CHR$(AF);
```

Okidata Printers

Microline 80	$349*
Microline 82-A	$499*
Microline 83-A	$749*
Microline 84	$1095*

*Free Shipping & Handling

Call: 1-603-525-6621

The Gollan Company
766 Middle Hancock Road
Peterborough, NH 03458

```
2360 NEXTJ,I
2370 IFCN>0THEN RN=RND(CN):I1=I(RN):J1=J(RN):CN=0
2380 IF(B(1)ORB(2))ANDP1(2,2)<3ANDP1(2,2)<>N AND RND(2)=1 THEN I
1=2:J1=2
2400 S1=-20
2410 RETURN
```

Instructions.

```
2500 CLS
2510 PRINTTAB(26)"Instructions:

   STAKE YOUR CLAIM is a two player game played on a game grid
of 25 squares (stakes) to be claimed. The object is to claim the
most squares."

2520 PRINT"
   A square facing an opponents's square may NOT be claimed, and
the first claim of each round may not be the center square.
2530 PRINT"
   When no more squares can be claimed, the round is over and
the player who has the most claims gains points determined by
the number of claims he has more than his opponent. The first
player to reach 7 points wins the game."
2540 PRINT"
Specifying "CHR$(34)"TRS"CHR$(34)" as one of the players names a
llows the"
2550 PRINT"user to play against the computer.            press
<ENTER>";
2560 I$=INKEY$:IFI$=""THEN2560ELSECLS:GOSUB1510:RETURN
2700 '
```

Winning music.

```
2710 FORK=1TO5:S$=AZ$(K):FORU=1TO10:AZ=USR(0):NEXT:NEXT
2800 RETURN
```

Data for lower case (HHH), sound (MMM), and winning music (WWW).

```
3000 DATA HHH,221,110,3,221,102,4,218,154,4,221,126,5,183
3010 DATA 40,1,119,121,254,32,218,6,5,254,128,210,166,4
3020 DATA 195,125,4,82,105,107,0,80,105,101,112,99,101,33
3050 DATA MMM,46,115,38,126,126,183,245,35,94,35,86,235
3060 DATA 241,200,61,200,61,245,86,30,1,29,35,78,35,62,9
3070 DATA 211,255,65,27,122,179,40,2,16,249,40,229,62,11
3080 DATA 211,255,65,27,122,179,40,2,16,249,32,228,24,213
3090 DATA 211,255,65,27,122,179,16,251,201,WWW,50,100,200
3095 DATA 50,120,180,60,120,180,30,60,90,25,50,100
```

Programmer's routine to test the sound generator.

```
3100 PRINT A$" ";
3110 A$=INKEY$:IFA$=""THEN3110
3120 S$="("+A$
3130 K=USR(0)
3140 GOTO3100
```

Another sound test routine.

```
3200 INPUT"NOTE NUMBER";U
3220 S$="!"+CHR$(U):FORX=1TO50:K=USR(0):NEXT
3240 GOTO3200
```

No-moves-left music.

```
3500 S$="<C(W(W<R<W":U=USR(0):FORTIM=1TO300:NEXT:FORX=1TO50:NEXT
:RETURN
3510 '
```

Set up sound strings.

```
3520 S$(1)="(W(P(WP'":S$(2)="(W(P(WP"+CHR$(218)
3530 S3$="!C!E!C(W<R<"+CHR$(198)
3540 PS$="(J+7(+<$"+CHR$(20)+"$($<+"+CHR$(20)+"+(+(7(+(7PJ"
3550 READA$:IFA$<>"WWW"THEN3550ELSEFORAZ=1TO5:AZ$(AZ)="":FORU=1T
O3:READK:AZ$(AZ)=AZ$(AZ)+CHR$(7)+CHR$(K):NEXT:NEXT:RESTORE
5000 '
```

Character graphics for title display.

```
5010 GL(1)=CHR$(166)+CHR$(179)+CHR$(147)+D$+CHR$(188)+CHR$(188)
+CHR$(182)
5020 GL(2)=CHR$(131)+CHR$(191)+CHR$(131)+D$+CHR$(176)+CHR$(191)+
CHR$(176)
5030 GL(3)=CHR$(190)+CHR$(179)+CHR$(185)+D$+CHR$(191)+CHR$(176)+
CHR$(186)
5040 GL(4)=CHR$(191)+CHR$(184)+CHR$(135)+D$+CHR$(191)+CHR$(178)+
CHR$(189)
5050 GL(5)=CHR$(191)+CHR$(179)+CHR$(139)+D$+CHR$(191)+CHR$(188)+
CHR$(190)
5060 GL(6)=CHR$(173)+CHR$(176)+CHR$(158)+D$+CHR$(176)+CHR$(191)+
CHR$(176)
5070 GL(7)=CHR$(158)+CHR$(131)+CHR$(173)+D$+CHR$(187)+CHR$(188)+
CHR$(183)
5080 GL(8)=CHR$(191)+" "+CHR$(170)+D$+CHR$(187)+CHR$(188)+CHR$(1
82)
5090 GL(9)=CHR$(191)+CHR$(179)+CHR$(187)+D$+CHR$(191)+CHR$(178)+
CHR$(189)
5100 GL(10)=CHR$(190)+CHR$(131)+CHR$(137)+D$+CHR$(187)+CHR$(188)
+CHR$(182)
5110 GL(11)=CHR$(191)+"  "+D$+CHR$(191)+CHR$(188)+CHR$(190)
5120 GL(12)=GL(3)
5130 GL(13)=CHR$(130)+CHR$(191)+CHR$(129)+D$+CHR$(184)+CHR$(191)
+CHR$(180)
5140 GL(14)=CHR$(24)+CHR$(170)+CHR$(157)+CHR$(144)+CHR$(152)+CHR
$(149)+D$+CHR$(24)+CHR$(24)+CHR$(186)+CHR$(181)+CHR$(178)+CHR$(1
76)+CHR$(181)
5150 RETURN
```

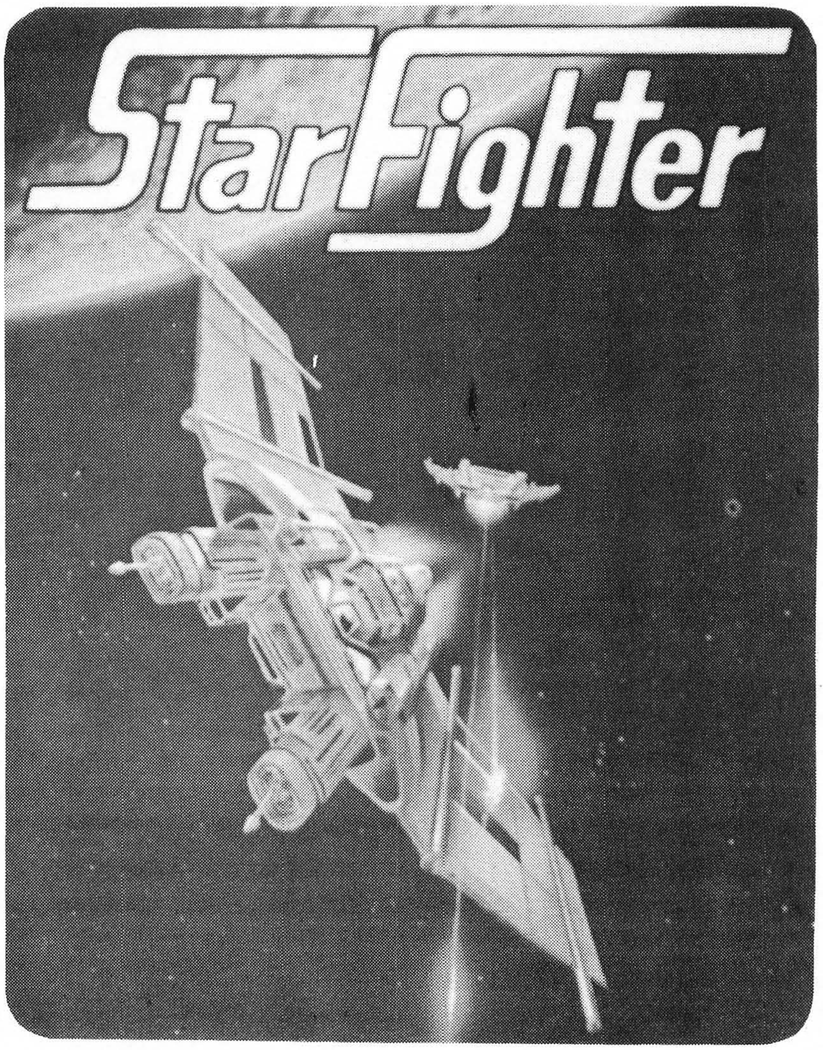
Starfighter

A review by Dave Albert and Alan J. Zett

by Sparky Starks (Adventure International). System requirements: 16K TRS-80® Model I or III, 32K with disk. Suggested retail price: disk — $29.95, tape — $24.95

You are returning from a successful foray in hyperspace, secure in the knowledge that the enemy spacecraft you have recently dispatched will guarantee your promotion when your performance is reviewed. As your computer scans the surrounding time/space continuum seeking other gravity fields, you savor your upcoming elevation in the socio-military scheme of things. Suddenly your combat computer begins to signal the presence of a gravity field within short hyperdrive range. Figuring to add one more notch to your beam weapon control key, you blast off on one final hunting spree before returning to Landbase Central.

Upon arriving at the spot in time and space occupied by the unknown gravity source, you shift to Combat mode and prepare to annihilate any Petro Resource Conglomerate craft in the vicinity. Suddenly your ship is rocked by a terrific blast and a message flashes across the top of your control display:



IDENTIFIED: INTERSTELLAR MINE PRC CRAFT ATTACK TO DESTROY

You also notice that colliding with that mine has left you only enough hypercharge to get to a nearby landbase. Your bank account is low; you had planned on doing some serious bounty hunting after your promotion. Now you must decide whether to head for a Landbase that will offer you enough bounty to purchase hyperdrive, and thus lose the credit for your kills and the chance for promotion, or to head for Landbase Central, get promoted, and not have enough hypercharge (or money) to do any more hunting. There really is no choice: You must collect the loot and the promotion will have to wait for another day. In a fit of disgust, you pound the Drive key before you prepare the craft for hyperdrive travel, and the last of your hypercharge dissipates into space. You are stranded.

Starfighter is a futuristic space combat simulation that requires skill, concentration, and a certain amount of luck. You are a military pilot for the Solar Galactic Authority (SGA). Your mission is to rid the universe of the minions of the Petro Resource Conglomerate (PRC). In essence, you are the bounty hunter of the future.

The SGA provides you with a brand new Starfighter spacecraft, equipped with sophisticated hypercharge weaponry and drive technology. They also provide you with an initial hypercharge field and a full tank of maneuvering fuel. After stern admonitions to attack only hostile spacecraft, you are catapulted into the space/time continuum and your career as a military fighter pilot begins. The duration of your career depends solely on your judgment and skills. Should you mistakenly obliterate friendly craft, the SGA authorities will demote you and generally make you feel lower than that creature from the annals of natural history, the worm.

New pilots are offered the choice of honing their skills in a special "space flight simulator" mode before actually trying their hand at real combat. In the simulator mode, the new pilot can learn the combat modes, visual displays, and movement characteristics of other craft by selecting an opponent from a list of all known spacecraft. This mode also helps to teach the pilot how and when to use the capabilities of his ship. In this manner, the *Starfighter* novice can learn how to anticipate or react to any situation instinctively. This can be a great help to the new pilot when computer information cannot yield positive identification of a craft or when circumstances require fast thinking and quick response for survival.

The program detail is well thought out and of high quality. The sound routines are complex; you won't find any simple beeps and clicks here. The visual graphics undergo contortions that would make a snake proud. The animation, while not smooth, is detailed and adds a feel of realism to the game. The instrument panel takes a bit of getting used to, but when mastered provides the resourceful pilot with a wealth of information.

In the combat mode there are two types of energy weapons; each is best for a specific situation. Other features include a targetting computer, target lock, maneuvering thrusters, target identifying computer, and in the event of an accident there is a mayday transmitter.

Figuring out which craft to destroy and which to ignore is another problem. The targetting computer, while helpful, often balks at rapid identification. Since the game provides many different types of spacecraft (enemy and ally), remembering which is on your side is difficult. Craft range in size from small mines to planet-smashing dreadnaughts. Distinguishing friend from foe is also complicated by the fact that friendly craft are sometimes stolen by pirates.

Another thing to remember is that you're not alone in the universe. There are other Starfighter craft searching for prey. Occasionally you'll find yourself face to face with one. How you react may well decide whether you live or die.

After you've destroyed your enemy (or ally), you can choose to search for more prey or dock at a landbase. There are eight such bases, however, since you're hopping around time as well as space, only some of the bases may be available to you. Landbase zero is provided for performance review and rank advancement; the other seven are reserved for bounty payments, refueling, tow tickets, or an overhaul.

The documentation for *Starfighter* is somewhat intriguing. It comes in the form of a 32-page Pilot Training Manual, complete with historical background, social and political briefing, hypercharge theory, instructions for using the Starfighter craft, and recommended procedures culled from the experience of other Starfighter pilots. It certainly reads like a military manual. It is informative, but also obscure and confusing. Plan on spending a few hours with it; it will take that long to digest it.

The only bugs encountered in the program were pretty ephemeral. Sometimes when trying to replenish the hypercharge supply the computer would reboot, thus wiping out hours of effort and play. There were also problems storing pilot records when foolishly taken with the urge to waste time eating or sleeping. But that problem probably originated in the disk drives, not in the program itself. Finally, there were some questions left unanswered by the documentation, including the mysterious extra hypercharge fields business, and the reluctance of the combat computer to identify friendly craft. But these problems actually can enhance the game . . . We all love an element of mystery.

All in all, *Starfighter* is quite an appealing game. Comparable to the ATARI® *Star Raiders* game, it has the excitement of a good arcade game combined with the intrigue of a good simulation. Should one progress through the ranks and reach the exalted rank of Star Lord, Adventure International informs us that there is a hidden message and password in the program that is the key to some sort of reward from the publishers. We suspect that this program will make the Top 10 games list this year.

DOSPLUS

How It's Better

By Alan J. Zett

Last month we listed the major differences between TRSDOS BASIC and DOSPLUS BASIC. This month we will deal with the major advantages to using DOSPLUS. Listed below are all of the new DOS features provided on *SoftSide*'s TRS-80® Disk Version. Although the version of DOSPLUS we distribute is a stripped-down one, there are a number of additional commands available from DOS and a random length file mode in BASIC.

New DOS Commands:

BUILD filespec

This feature allows you to create a DO file. A DO file can contain a series of command lines up to 63 characters in length. These command lines can contain either DOS commands or BASIC program lines. The primary function of this feature is to allow you to enter a BASIC program right from power-up. You can set the number of files, memory size, and execute the program without operator input, as well as load special drivers from DOS. If you don't specify a file extension, /BLD is automatically assumed.

To create a DO file, type in a line of commands and then press ENTER. DOSPLUS will then prompt you for the next line. When you have finished entering commands, press the BREAK key in response to the prompt and the file will be saved on drive 0.

CLEAR

This command clears all user memory above 22272 (5700H) while doing a quick memory check. To execute this feature, type CLEAR and press ENTER.

CONFIG (parameter1,parameter2, parameter3,etc.)

This command allows you to customize DOSPLUS to your specific needs. Legal parameters are:

TRKS = nn — Tells DOS the number of tracks on the system disk. This will cause DOSPLUS to make backups using the specified number of tracks.
STEP = nn — Sets the head step rate. Be cautious when using this parameter. Legal step rates are 6, 12, 20, 30, and 40 milliseconds.
GPHON — Enables graphics to be sent to the printer.
GPHOFF — Changes graphics to asterisks when printing.

CREATE filespec (NRECS = nn, LRL = nn)

NRECS is the number of records to allocate.
LRL is the logical record length of these records.

This feature is used to create a file and pre-allocate disk space. The advantage of using CREATE is that it will speed up file access time during disk I/O.

DEVICE

This function will display on the screen all I/O devices and their driver addresses. Some devices are *KI (keyboard), *DO (video), *PR (printer), *RI (RS232 input), and *RO (RS232 output). Addresses given are in hexadecimal.

DO filespec

This feature begins automatic execution of a file created with the BUILD command. If no extension is given, /BLD is assumed. DO is legal during an AUTO command.

FORCE devicespec TO devicespec

Where devicespec is any legal device name. See DEVICE command for legal names.
This function allows the user to route I/O between various devices. Its primary use is to route printer output to the screen, etc.

FORMS (P = nn,L = nn,W = nn, S,LF,T)

P = nn — Maximum number of lines per page.
L = nn — Number of printed lines per page.
W = nn — Number of characters per line.
S — Directs all output to a serial printer.
LF — Will generate a linefeed with a carriage return.
T — Performs a top-of-form on the printer.

This function controls the print driver and will allow you to direct the output to either the serial or the parallel printer ports, as well as tell the printer the size of the paper you are using.
Default values: P = 66, L = 60, W = 132, S = off, LF = off, T = off.

PAUSE remark

Where remark is an ASCII character to display on the screen to show that the computer is waiting for a response.
This function stops execution of a DO file and waits for operator input.

RS232 (parameter1,parameter2, parameter3,etc.)

This feature allows you to display

and alter the switch settings on the serial interface board. Parameters are:

BAUD=nnnnn — Baud rate. Can be 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, or 19200.
WORDS=n — Bits per word. Can be 5, 6, 7 or 8.
STOPS=n — Number of stop bits. Can be 1 or 2.
PE — Parity enable.
PI — Parity inhibit.
EVEN — Even parity.
ODD — Odd parity.
BREAK — Not ready to receive data.
RTS — Request to send.
DTR — Data terminal ready.
NOWAIT — Tells RS232 drivers not to wait for a byte.

Typing RS232 without parameters will display the current settings of the serial interface, if one is installed.

Enhanced BASIC command:

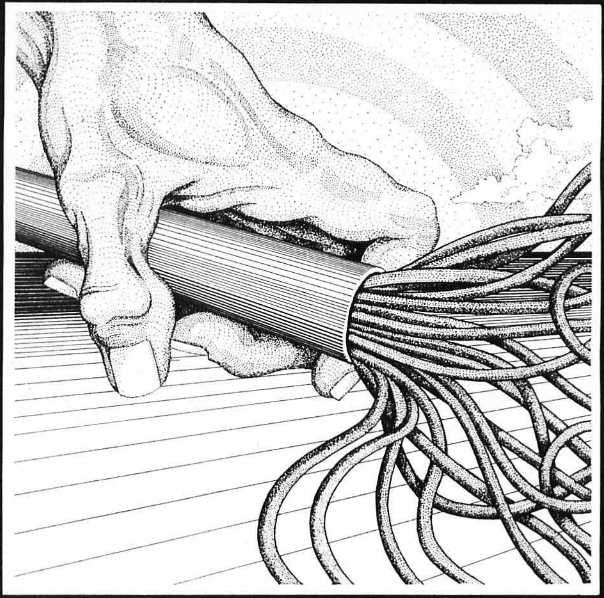
OPEN "R",buffer,filespec,LRL

LRL is the logical record length. When omitted, LRL=256 (normal). Special note: When a file is opened with a specific record length, attempts to open the file at a later date will cause errors to occur if a different length is specified.

The OPEN "R" command has now been expanded to allow you to specify the number of bytes in a record. You may now span the track and sector boundaries of a disk file. As an example, if your record is 45 bytes long you will be able to put five records on one sector. The 31 bytes that remain on this sector will be used by the sixth record with the remaining 14 bytes being the first 14 bytes on the next sector.

In TRSDOS Disk BASIC, you had to calculate sub-records yourself. With LRLs, each sub-record can be treated as an entire record. This eliminates the need for lengthy calculations. For example: To get the fifth record, do a GET 1,5 and you will get the fifth record. Note also that when LRL is not specified, all normal TRSDOS files will still function properly.

EXPAND YOUR *TRS-80 CAPABILITIES



THE SHADOW

What secrets lurk deep within the heart of your microprocessor? Only THE SHADOW knows. It is a machine language program which disassembles and examines program instructions from any part of your computer memory. THE SHADOW allows you to single-step through your *TRS-80 Model I or III ROM. Disk only, specify Model I or Model III.

SUPERKEYS

With this utility, commonly used BASIC functions and statements can be entered with a *single keystroke.* Any of 225 characters can be designated into every key. SUPERKEYS also offers a "relocate" feature which allows for single-line relocation and renumbering. Disk for Model III only.

AOS EXPANSION PACKAGE 1

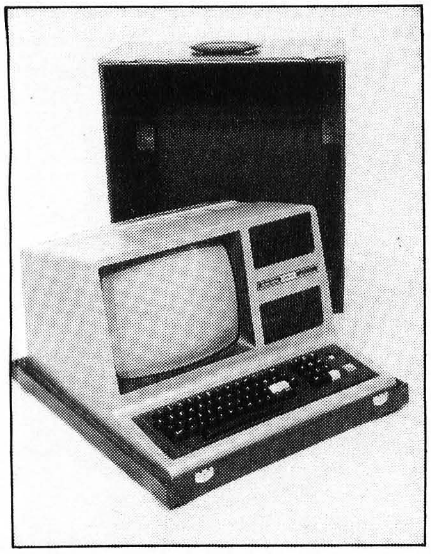
Two separate programs are included in this expansion package which help you get more out of your *TRS-80. VARKEEP is a variable-passing utility which can SAVE variable values and pass them to other programs, allowing you to perform true program chaining. SCREENPACKER is a powerful graphics utility which allows you to "draw" images, produce large graphic letters, and SAVE them on disks. Disk only, specify Model I or Model III.

Now available at your local software retailer, or call 1-800/348-8558 to order. (Indiana residents, call 1-219/879-4693.) MasterCard and VISA accepted.

Advanced Operating Systems
450 St. John Road
Michigan City, IN 46360

*TRS-80 is a registered trademark of Radio Shack, a division of Tandy Corp

NEW PRODUCTS

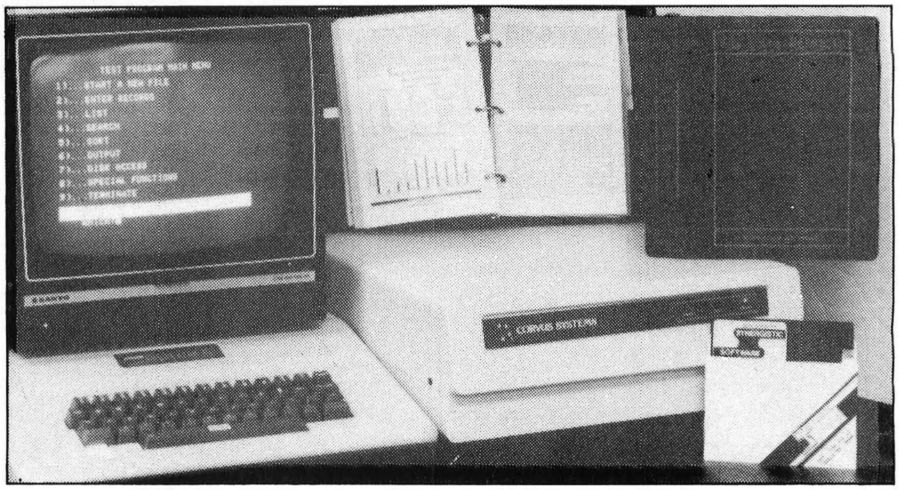


Computer Case Company
5650 Indian Mound Court
Columbus, OH 43213
(614)868-9464

The COMPUTER CASE COMPANY of Columbus, Ohio has announced the addition of five new cases to their line of COMP-CASES for microcomputers and related equipment. The new cases include the AP104 (retail price: $139) for the new Apple III. This case holds the computer, two additional disk drives and a Silentype printer. The AP105 (retail price: $99) holds a monitor such as the Sanyo or Leedex 13''. The RS204 which retails for $129 is for the TRS-80® Model III and the RS205 retailing for $99 is for the Radio Shack Color Computer. The P403 is for the Epson MX80 and MX70 printers and retails for $99.

These cases provide not only portability but a convenient method of storage free from possible damage and dust accumulation. By replacing and locking the lid the computer and software are protected from possible inadvertent damage or failure due to repeated connecting and disconnecting.

The cases are constructing of luggage material with hard sides, padded handles, brass hardware and key locks. Rubber pads provide furniture protection and steel lugs on the bottom protect the case when transporting. The outside is covered in high quality scuff resistant textured vinyl in rich brown. The tops are easily removed so that the equipment can be operated without removal from the case. Provisions are made for cords to exit the case even when the top is on and locked, to provide convenient security without the need to disconnect electrical cords and cables. Storage space is also provided for manuals, cords, working papers, and supplies within each case.



Synergistic Software
5221 120th Avenue SE
Bellevue, WA 98006

A new, faster, more complete data management package, The Data Reporter, is now available for the Apple™ II computer using floppy or hard disk drive. The package includes a powerful database, a report generator, a plotter/analyzer program, calculator capabilities, and a variety of utilities, all designed to work together on common data files. The Data Reporter is a rapid response, general purpose data management system that can be automatically customized to your particular data storage, access, and manipulation requirements. The programs will create, under your direction, any number of new special purpose data management programs. You can make your own database, inventory control, accounts receivable, sales analysis, bibliography, memo programs, etc. The set up can easily be changed at any time, and all data files can be reformatted without reentering data. The sophisticated text editor lets you design and print letters, reports, documents, contracts, etc. with ease. Output commands for printer control, graphs, statistics, labeling slide shows, and more are provided. No special data input is required as each program can read any data file created with The Data Reporter. Requires 48K, Applesoft, at least one disk, and DOS 3.3. Hard drive or floppy drive versions $220.00 from Synergistic Software, 5221 120th Avenue SE, Bellevue, WA 98006, 206-226-3216.

The Harvard Newsletter On Computer Graphics
730 Boston Post Rd
P.O. Box 89
Sudbury, MA 01776
(617)443-4671

Free Guide to Computer Graphics Markets, Companies, and Business activities is available.

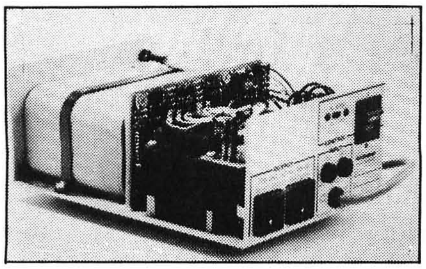
Computer graphics is a key to industrial and business productivity improvement. As a result, computer graphics markets are thriving, and companies and businesses in the field have become of keen interest to corporate planners and managers, investors, and enterpreneurs. As a result, The Harvard Newsletter On Computer Graphics, published under the auspices of the Laboratory For Computer Graphics at Harvard University, has made available a free guide to computer graphics markets, companies, and business activities.

The guide contains succinctly written descriptions of market intelligence, recent public offerings, enterpreneurship, and mergers/acquisitions, including, wherever appropriate, the name of each subject company, including mail address, person to contact, and phone number. Sources outside the U.S. are included as well.

''The intent is to help noncomputer graphics experts in business to track developments in the field,'' says Stanley Klein, Publisher of the Guide and also publisher/editor of The Harvard Newsletter.

To receive a free copy, write on letterhead, enclosing a self-addressed envelope, prestamped (.37 cents U.S.; $1.20 elsewhere), to Markets CG Guide, c/o The Harvard Newsletter On Computer Graphics, Service Dept. — P.O. Box 89, Sudbury, MA 01776.

NEW PRODUCTS

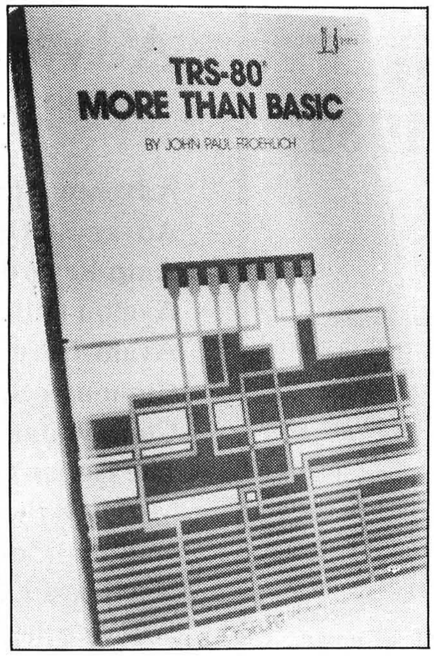


CUESTA SYSTEMS, INC.
3440 Roberto Court
San Luis Obispo, CA 93410

CUESTA SYSTEMS, Inc. has provided power 'insurance' for users of microcomputers and other electronic equipment with the introduction of the DATASAVER™, a low-cost AC power backup unit.

Like a life insurance policy, DATASAVER safeguards the time and information invested in high technology systems by protecting electronic equipment against unexpected power failures or sudden surges that can cause many systems to fail.

As a result, the loss or accidental modification of temporary or real-time data now can be virtually eliminated. DATASAVER also guards against program or disk drive crashes during critical operating modes. And the need to manually reinitialize computers after a loss of power has become a thing of the past.



Group Technology, Ltd.
P.O. Box 87
Check, Virginia 24072
(703)651-3153

TRS-80®, More Than BASIC, a 220-page textbook by John Paul Froelich, presents a monitor program that makes the TRS-80® Model I or III microcomputer into a development system. Conversion of the TRS-80® can be done by loading object code from a cassette or diskette or by replacement of the BASIC read-only memory (ROM) unit. The monitor then executes valid instructions or commands and it flags errors. Over 26 commands are available to enable one to program in Z-80 mnemonics. Complete documentation is given.

Detailed descriptions of the hardware of both Model I and Model III systems orient the reader for making the necessary connections to implement the monitor. A description of the FROLIC monitor follows along with the commands available to the user. Hardware that allows the inclusion of a single-stepping feature forms an attractive additional command to aid in system development.

Chapter 3 tells how to obtain written text useful in tracing system development. Chapter 4 describes the hardware for programming erasable read-only (EPROM) devices, a state-of-the-art programmer with features not normally found on commercial equivalents at one hundred times the cost. The source code for driving the programmer for programming the most popular EPROMs is also given. The appendixes provide all the source code for the monitor; the user can thus alter the monitor to customize it for individual applications. The book permits and encourages the reader to acquire and develop skill in programming a microcomputer in the instruction code of the microprocessor. CHAPTER TITLES: More Than BASIC, The Monitor, Hard Copy from the Monitor, PROM Programmer. Appendixes: Command Sequence Table, References, Hardware and Software Supplier, Source Listing for FROLIC Monitor for Model I TRS-80®, for Model III, for the 2708 EPROM, for the 2716 EPROM, for the 8755 EPROM.

Electronic Specialists, Inc.
171 South Main Street
Natick, MA 01760
(617)655-1532

Direct Plug Isolators

ELECTRONIC SPECIALISTS expands their patented ISOLATOR Line to include units that plug directly into the wall socket. Designed for installations that do not require extension cords, the Direct Plug ISOLATORS provide the same equipment interaction isolation and power line protection as their popular line cord ISOLATOR series. A convenient retention screw prevents accidental withdrawal from the wall socket.

Direct Plug ISOLATORS can accommodate a total 1875 watt load, with up to 1000 watts per socket. A high capacity Spike/Surge Suppressor is designed into each unit.

Direct Plug SUPER ISOLATOR (Model DP-SIS032) provides 2 super-isolated channels for $96.95.

Automated Simulations
P.O. Box 4247
Mountain View, CA 94040

Dragon's Eye, an EPYX game from Automated Simulations, Inc., is now available for the ATARI® 400 and 800 computers.

Dragon's Eye, an overland fantasy role playing game, challenges the player to find a magical gem, the Dragon's Eye, in 21 game-days (approximately half-an-hour playing time).

The Eye is hidden somewhere within one of the seven provinces, and the player must find it and return it to Fel City, where his journey began.

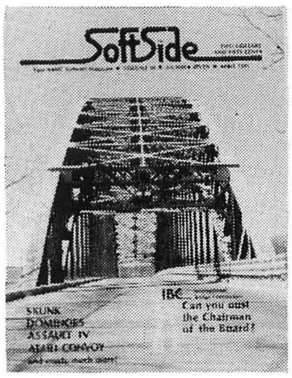
The player chooses one of 16 characters and gains a set of magical abilities, such as healing, flying, time travel and teleport. Which spells he gets are different each time he plays. He is equipped with his choice of four swords, a bow and arrows, and magic bolts.

While searching, the player sees a detailed map of the provinces on the screen, along with his location, strength health, and other information.

The player encounters dragons, bats, vampires, ghosts, golems, serpents, skeletons and other monsters. He can choose between 13 commands, from firing an arrow and fighting with sword to casting magic spells and searching for hidden paths.

When a battle is engaged, fully animated graphics display the action between player and beast.

Dragon's Eye is available on cassette for the ATARI® 400/800 (32K) and PET (32K), or on disk for the APPLE™ (48K with ROM Applesoft), and ATARI® 400/800 (32K), from Automated Simulations, Inc., P.O. Box 4247, Mountain View, CA 94040. The suggested retail price is $24.95.



Back Issue of the Month: April 1981

Eleven months ago, *SoftSide* ran *International Bridge Contractors* for the TRS-80®. The ATARI® version of that program is this month's translation contest winner.

The April '81 issue also includes: *Battle at Sea* for the Apple™, *Strategy Strike* and *Convoy* for the ATARI®, and *Assault IV* for the TRS-80®.

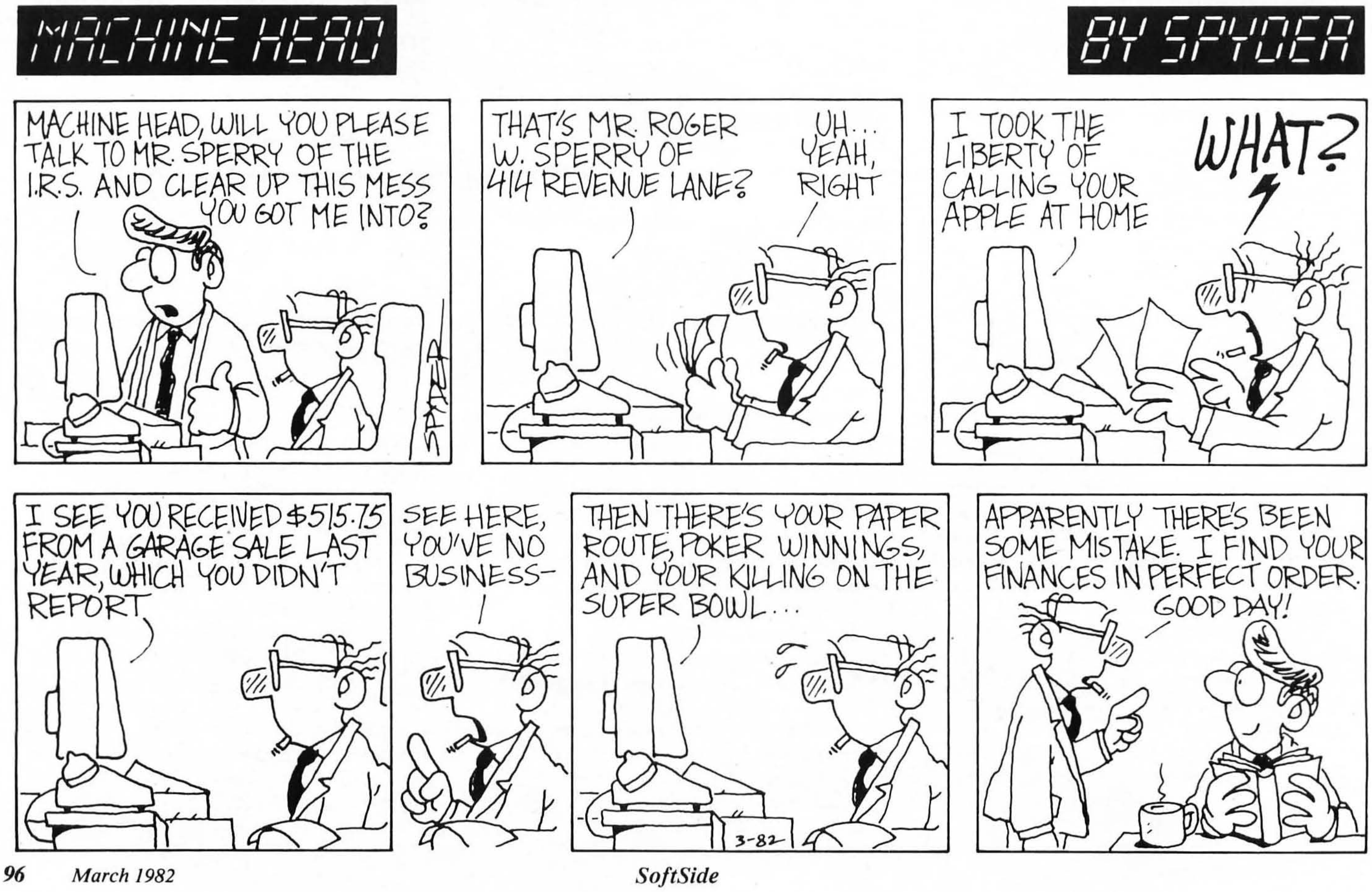
We also have a couple of old favorites: *Darts* for the Apple™ and *Dominoes* (three versions in one program) for the TRS-80®, and *Flags* (tutorial game with sound) for the Apple™.

While they last, copies of the April, 1981, issue of *SoftSide* are available for a mere $3.50. For a bit more we'll include a cassette ($9.95) or disk ($14.95) containing all the programs for your computer that appear in the issue. Remember to specify what computer you use when ordering the magazine and media combinations. Order now and prepare yourself for yet another magazine full of top-notch software for you and your family to enjoy.

See our Back Issues ad on page 80 for ordering information.

Advertiser's Index

Adventure International	**Cover III**
Advanced Operating Systems	**49, 93**
Amp Recordings	**54**
Avalon Hill	**Cover II, Cover IV**
Avant-Garde Creations	**41**
Computer Services Corp. of America	**4**
Continental Adventures	**62**
The Gollan Company	**88**
MICRO, the 6502/6809 Journal	**1**
National Computer Shows	**32**
Northeast Expositions	**56**
***SoftSide* Selections**	**7, 9, 12, 13, 16, 17, 28, 31, 54, 57, 64, 80, 81, 87, 91, 96**
Spectrum Computers	**73**


MACHINE HEAD
BY SPYDER
MACHINE HEAD, WILL YOU PLEASE TALK TO MR. SPERRY OF THE I.R.S. AND CLEAR UP THIS MESS YOU GOT ME INTO?
SPYDER
THAT'S MR. ROGER W. SPERRY OF 414 REVENUE LANE?
UH... YEAH, RIGHT
I TOOK THE LIBERTY OF CALLING YOUR APPLE AT HOME
WHAT?
I SEE YOU RECEIVED $515.75 FROM A GARAGE SALE LAST YEAR, WHICH YOU DIDN'T REPORT
SEE HERE, YOU'VE NO BUSINESS-
THEN THERE'S YOUR PAPER ROUTE, POKER WINNINGS, AND YOUR KILLING ON THE SUPER BOWL...
3-82
APPARENTLY THERE'S BEEN SOME MISTAKE. I FIND YOUR FINANCES IN PERFECT ORDER.
GOOD DAY!

BOUND TO PLEASE: SoftSide's Deluxe Edition

Twelve issues of SoftSide — October 1980 to September 1981

SoftSide is pleased to announce this limited edition: twelve issues sewn into blue buckram. This sturdy library bound collection will be available in limited quantities, so order yours now.

The SoftSide Deluxe Edition — $75.00
Send a check or use your MasterCharge/VISA credit card.
To order write: **SoftSide Publications**
6 South Street
Milford, NH 03055

The ELIMINATOR

FEATURING
SPECTACULAR
SUPER SHARP
GRAPHICS
& SOUND

"THE BEST ARCADE TYPE GAME I'VE SEEN . . . GREAT FIRE POWER!"

ROB McCONNELLI,
ARCADER

"NOTHING MATCHES ELIMINATOR FOR SHEER FUN. I CAN'T STOP PLAYING!"

PAT HENDERSON,
ARCADER

Your the pilot of The Eliminator, a space fighter of the Defender Class. It has lighting fast response and is armed to the teeth with awesome firepower.

But you're outnumbered! And your attackers are keen. Your only options are victory or a grave in space.

APPLE VERSION BY JOHN ANDERSON
TRS-80 VERSION BY WAYNE WESTMORELAND & TERRY GILMAN

APPLE 2 · 48K DISK (DOS 3.3 REQ'D.)	042-0134	$29.95
TRS-80 · 16K TAPE MODEL 1 OR 3	010-0134	$19.95
TRS-80 · 32K DISK MODEL 1 OR 3	012-0134	$24.95

Adventure INTERNATIONAL © 1981
A DIVISION OF SCOTT ADAMS, INC.
BOX 3435, LONGWOOD FL 32750
(305) 862-6917 (QUESTIONS)
ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172 (ORDERS ONLY PLEASE)
SHIPPING & HANDLING ARE EXTRA, PRICES SUBJECT TO CHANGE WITHOUT NOTICE
WRITE FOR OUR FREE 150 PROGRAM CATALOG

ART © 1981 · DON DIXON