

OCTOBER 1981

SoftSide™

THREE DOLLARS

VOLUME V • Your BASIC Software Magazine • NUMBER ONE

I-\$TRING
Produces **Envyrn™**



SoftSide DV, the magazine of the future, is here!

If your computer could pick a magazine, wouldn't it prefer one in its own language? Now there's one available.

SoftSide DV is an enhancement of the **SoftSide** you have in your hands.



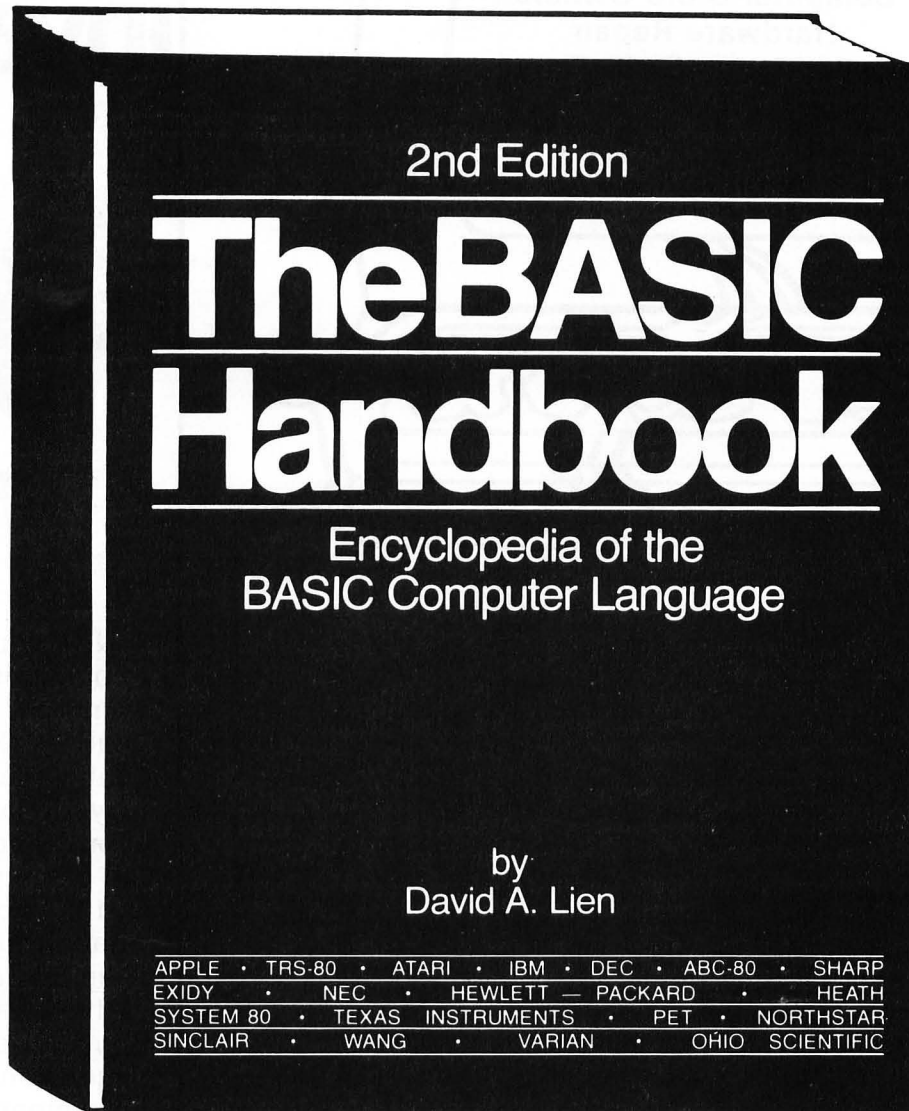
Complete programs of every conceivable type, ongoing data bases, modified languages and I-String, the video simulator — all of these are part of **SoftSide DV**. Feel as though you're missing something? You are! But, you needn't any more.

Introductory offer — if you subscribe to **S-80 SoftSide DV** before September 30 at the regular price of \$125 per year (\$10.42 per issue), you'll receive the September 1981 disk, including **NEWBASIC**, absolutely **FREE!** Or, if you'd like to look us over, try the September issue only, for \$12.95

For your convenience, we offer an installment payment plan for Visa and Mastercard holders: You pay only \$32.50 per quarter (includes a \$5 billing charge). For orders outside the USA, please add \$18. If you currently receive **SoftSide**, we'll credit the remainder of your subscription to your **SoftSide DV** order.

We're developing **SoftSide DV** for one system at a time. The S-80 version is currently available with the Atari and Apple editions soon to follow. In the meantime, new subscribers to the disk version for Atari and Apple will receive a \$10 **SoftSide Selections** gift certificate in addition to the monthly magazine and disk containing **BASIC** program listings.

Expand Your Computer's Vocabulary!



The BASIC Handbook has never been this complete.

The Expanded Second Edition gives you over twice as much information as the First Edition, explaining nearly 500 BASIC words. The handbook features special sections on Disk BASIC, TRS-80 Extended Color BASIC, Atari BASIC, Acorn Atom BASIC, Tektronix BASIC and "Converting Programs From One Computer For Another."

The computer industry has experienced tremendous change in the last three years. Hundreds of new computers have been introduced since **The BASIC Handbook** was released in 1978. The Second Edition meets the challenge head-on, documenting every significant BASIC word used by every BASIC-speaking computer.

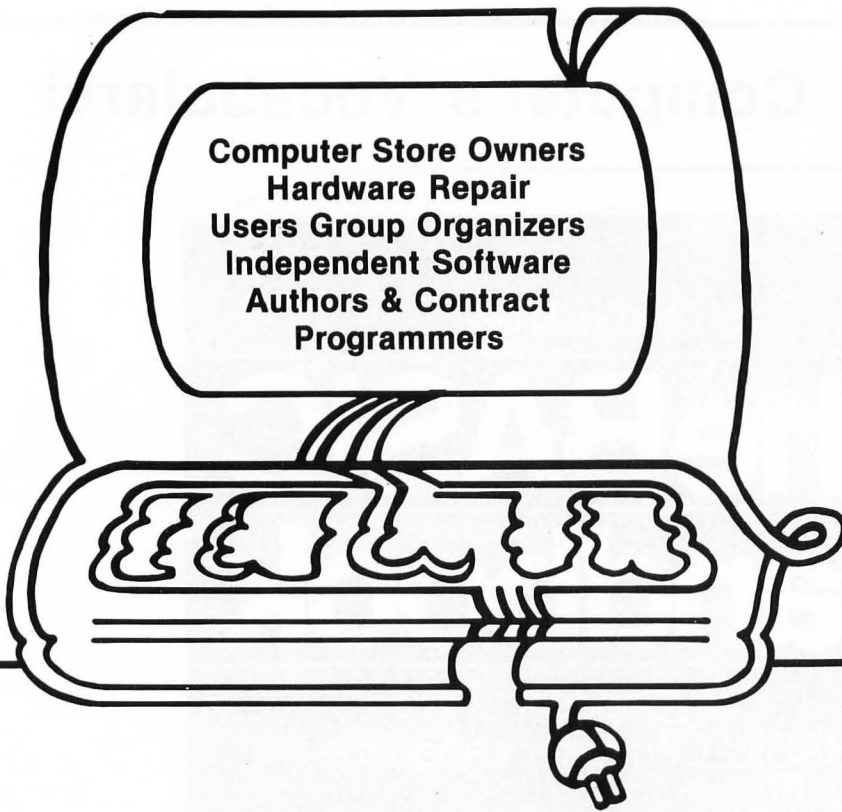
This new Edition makes program conversion easy. Its widely acclaimed feature, "If Your Computer Doesn't Have It" has been expanded. Each BASIC word is alphabetically listed, with **Test Programs**

and **Sample Runs. Variations in Usage** combine with **Alternate Spellings** to totally cross-reference each BASIC word.

INTRODUCTORY SPECIAL — 10% OFF!
The BASIC Handbook by David Lien
(Reg. \$19.95) \$17.95

(Please allow six weeks for delivery)

SoftSide™
Selections 
6 South Street Milford NH 03055



Computer Store Owners
 Hardware Repair
 Users Group Organizers
 Independent Software
 Authors & Contract
 Programmers

Do you offer products or services that should be of interest to the computer hobbyists in your area? OR, do you possess the skills to repair computers, offer consultation, organize users groups, but don't really know how to get started? Well, the first step is locating potential customers, efficiently and inexpensively. We at **SoftSide** can help.

Tap a responsive market with **SoftSide's** mailing lists. We've compiled a list of close to 80,000 names of individuals who have bought computers, peripherals and software or who have inquired about these items through ads in national magazines. Now you can access selections from this list to advertise the services you now provide or are considering.

Most sources of lists require a minimum order of 5000 names. **SoftSide's** list is available to you in groups of 1000, arranged by zip code, for \$100.

We'll send you the list of people in your area on 4-up cheshire labels or on pressure-sensitive labels (for an additional \$1 per 100), ready for you to mail with your promotional flyer to potential customers. Just let us know which area you wish to cover with your mailing.

Our lists are a good start toward advertising locally — you'll reach the people who will be responsive to your offers. It's more efficient than the Yellow Pages, less expensive than newspapers.

We'll put you in touch with the right people. Between our connections and your talents, we can make a great combination!

For a distribution of available groupings of names in your area, send a self-addressed stamped envelope to:

SoftSide Mailing Lists
 6 South Street
 Milford, NH 03055

Labels are authorized for one-time use only.

ALIEN INVASION



Now you can continue to play this popular game even after you run out of quarters! Hi-Res arcade quality game pits the player against an elusive alien flying saucer. Use the game paddle to steer and fire your laser weapon. But beware: the alien aircraft moves fast and in a random way. Sound effects add another dimension to the enjoyment of this amusement. (Sound requires external amplifier.)

S-80, 16K,
 Machine Language
 Cassette \$9.95
 S-80, 32K,
 Machine Language
 Disk \$14.95



SoftSide™

Volume V — Number One

PUBLISHER
Roger Robitaille Sr.

MANAGING EDITOR
Randal Kottwitz

PROGRAMMING EDITOR
Jon Voskuil

EDITORIAL DEPARTMENT
Scott Adams
Rich Bouchard
Dean F. Macy
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Ed Ulmor
Alan J. Zett

PRODUCTION MANAGER
Nancy Lapointe

PRODUCTION DEPARTMENT
Lynda Fedas
Tom Stanton

ADVERTISING
Nancy Wood

SUBSCRIPTIONS
Diana Bishop
Cindy Schalk

STAFF
Kathleen Boucher
Philip Brown
Pam Demmons
Stephen Justus
Nancy Macy
Doris Miller
David Robitaille
Christine Spade
Anmar William
Gary Young



This month's cover features an artist's rendering of the I-String concept. A complicated city (top) can be codified by Envyrn™ (bottom) to produce an Envyrnment™ (center). Illustration by Bill Geise.

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, \$30.00/12 issues. USA First Class APO, FPO, Canada, 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), \$36.00/12 months. All remittances must be in U.S. funds. Mail subscription inquiries to SoftSide Publications, P.O. Box 68, Milford, New Hampshire 03055. Entire contents copyright 1981. SoftSide Publications. All rights reserved.

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

If you have not received your November issue of SoftSide by Nov. 6, contact SoftSide Publications, 515 Abbot Drive, Broomall, PA, 19008 or call 1-800-345-8112 (In PA call 1-800-662-2444).

FEATURE

- 47 I-String Produces Envyrn™**
A revolutionary breakthrough in entertainment software. . . . Roger W. Robitaille, Sr.
- 48 About the Centerfold** Staff
- 49 Envyrn™ Preliminary Editor**
The first code to be unveiled Rich Bouchard
- 52 Envyrn™, Its Marketing**
A fresh approach to a unique product. Roger W. Robitaille Sr.
- 53 Envyrn™ GAMEPLAY/BAS**
Instructions for S-80 DV Rich Bouchard

ARTICLES

- 76 Take Apart: Atari Quest**
The technique explained Alan J. Zett
- 90 Reviews**
Apple II User's Guide Jon Voskuil
Pool 1.5 Rich Bouchard
Orchestra 80 Robb Murray

APPLE, S-80 AND ATARI PROGRAM

- 21 Leyte**
Step back to 1944 and World War II Victor A. Vernon, Jr., Alan J. Zett

APPLE AND S-80 PROGRAM

- 55 Arena of Octos**
Can you defeat the creatures of eight? Al Johnston, Steve D. Kropinak

APPLE PROGRAMS

- 40 Developing Database**
The long-awaited complete Apple listing. Mark Pelczarski
- 54 Super Dairy Farming**
Apple DV Premieres Jean H. Anderson
- 72 Plotting the Yang/Yin Symbol**
Esoteric computer applications David Delli Quadri

ATARI PROGRAM

- 78 Character Generator**
An artistic utility Alan J. Zett

S-80 PROGRAM

- 35 ABM Command**
Defend the Eastern Seaboard with your S-80. Arnold E. van Beverhoudt, Jr.

DEPARTMENTS

- 05 Editorial** Randal Kottwitz
- 06 Input** From Our Readers
- 07 Hints and Enhancements** From Our Readers
- 10 Outgoing Mail** Randal Kottwitz
- 10 About This Issue** Randal Kottwitz
- 11 Calendar** Kathleen Boucher
- 13 My Side of the Page** Lance Micklus
- 17 The Sensuous Programmer** "J"
- 54 Machine Head** Spyder Webb
- 84 What's New** Dean F. Macy
- 85 Potpourri** Dean F. Macy
- 89 Bugs, Worms, and Other Undesirables** Editors

*TRS-80, Apple, and Atari, are registered trademarks of The Tandy Corporation, The Apple Computer Company, and Warner Communications.


ATARI® SOFTWARE PIRACY: THIS GAME IS OVER.

ATARI® has led the industry in the development of video games such as ASTEROIDS™ and MISSILE COMMAND™. The outstanding popularity of these games has resulted from the considerable investment of time and resources which ATARI has made in their development. We appreciate the worldwide response from the videophiles who have made our games so popular.

Unfortunately, however, some companies and individuals have copied ATARI games in an attempt to reap undeserved profits from games that they did not develop. ATARI must protect its investment so that we can continue to invest in the development of new and better games. Accordingly, ATARI gives warning to both the intentional pirate and to the individuals simply unaware of the copyright laws that ATARI registers the audiovisual works associated with its games with the Library of Congress and considers its games proprietary. ATARI will protect its rights by vigorously enforcing these copyrights and by taking the appropriate action against unauthorized entities who reproduce or adapt substantial copies of ATARI games, regardless of what computer or other apparatus is used in their performance.

We ask that legitimate software developers cooperate with us to protect our property from any form of software piracy, imitation or infringement. ATARI is currently offering copyright licenses for a limited number of its games to selected software developers. If you happen to be selling a software product which performs a game similar to any ATARI game (such as a game created for a home computer), please contact us immediately. Write to the attention of: Patent Counsel, ATARI, Inc., 1265 Borregas Ave., Sunnyvale, Calif. 94086



 A Warner Communications Company

© 1981, ATARI, INC.



Editorial

Randal L. Kottwitz

Amuse — a verb — To occupy in an agreeable, pleasing, or entertaining fashion.

Computers as amusement — a much more difficult phrase to define. As the computer weaves itself ever more firmly into the fabric of our lives, we are discovering our former definitions for work, education and amusement no longer seem to fit so well. As I conceive of my computer in relation to its amusement value, I realize that I am equally as entertained by the ease with which I can enter and edit this column with my word processor as I am by battling it out with the Klingons in any version of Star Trek. Work becomes play.


Educators are breaking into new frontiers with complicated computer simulations which allow their students to experience the actual decisions of life rather than depending on background information, already outdated. The creative processes of the office are encouraged through complex computer controlled games. The military is installing "arcade style" simulators on its bases to encourage improvement of hand-to-eye coordination. All of these might have been considered strictly "amusement" only a few years ago and are now serious applications to ease and improve the process of learning.

Certainly, all of these uses of the computer do "occupy in an agreeable, pleasing or entertaining fashion," but combine that occupation with an important and productive function. The hidden word with the most important impact there is "productive." Suddenly we find that our amusement can produce very practical results. Sociologists have predicted that the future holds a great deal more "leisure time" than that to which we've become accustomed. My impression of that prediction has always been that such an increase would lead to a lessening of the personal productivity level. I am now forced to reconsider the idea in terms of this melding of "leisure" and "working" time, as precipitated by the "information/computer revolution."

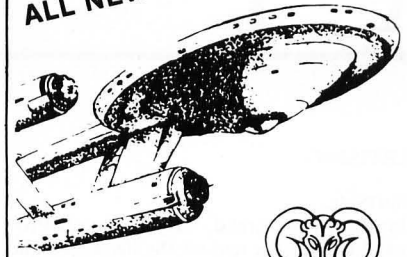
This month, **SoftSide** brings to its pages a concept in programming that may substantially speed the "crossing of the lines" from entertainment to employment to education. The I-String concept driving **Envyrn™** makes the

computer substantially more "user friendly." Vast amounts of complex information can be stored in easily accessible units for later recall. Those units may then be assembled in any order over a virtually unlimited area. Physical interrelationships may be easily simulated and tested. Information may then be observed with three degrees of accuracy, allowing quick overscan of large volumes or the study of minute detail in small quantities. These concepts may seem rather vague to you — don't feel alone! In the months I-String has been developing at **SoftSide**, the staff has looked to the project with glazed eyes. We were all aware that we were working on something of great importance, but the magnitude of its influence and applications has only recently become crystal clear.

I-String is a concept — an important one. The technique is one which can make child's play out of storage, manipulation and observation of data. **Envyrn™** is the tool which will result, based on the concept of I-String. With that tool, educators, businessmen and entertainers will be able to easily present their "audience" with complex material in a simplified manner. Students will be presented with universes to challenge their imaginations and then allowed to create ones of their own. Detailed inventories will be inspected by use of a map of the storage area on a screen. Communities will be planned by allowing the physical elements of the proposed plans to interrelate, with the computer exposing possible problems, years before they might otherwise be detected. Game authors will have a new world opened to them. Never before has this approximation to realism been available and the ease of application will allow anyone to write complex entertainment software without complex programming skills.

All of these things will "occupy in an agreeable, pleasing or entertaining fashion," yet make productivity and use of information an easy option. The day is approaching when the difference between work and play is only a "point of view." Read the material on I-String and **Envyrn™** in this issue carefully, you may then agree that the term "amuse" may not be so easily defined for much longer. 

ALL NEW VERSION!



STAR TREK III.5

Now with Sound Capability and Increased Speed of Execution.

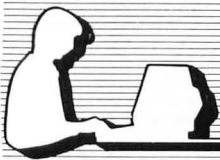
by Lance Micklus

You are in command of the starship Enterprise and her complement of 371 officers and crew. You must enter and explore the Omega VI region of the galaxy with its 192 quadrants containing star systems and planets (a few of which are habitable). Astronomical hazards such as pulsars, Class 0 stars, and black holes are known to be present, so the utmost care is needed.

Star Trek III.5 includes, playboard 8 by 8 by 3 quadrants, weapons system of Phasers and Photon Torpedos; Warp and impulse power systems; Science and Ship's computers; Long and Short Range sensors; Damage Control and Status reports; and 20 Klingon battle cruisers, and 100 stars, planets, black holes, and pulsars.

Atari 32K Tape \$19.95
S-80 16K Tape \$14.95
S-80 32K Disk \$15.95





ADVERTISING

Dear **SoftSide**,

As far as I'm concerned, "The Sensuous Programmer" has hit the nail on the head with the comments made in the July issue concerning "tacky" programs. When it comes to buying advertised programs, nothing turns me off faster than an advertisement replete with spelling and grammatical errors.

As a prospective mail-order purchaser, I must rely on advertisements published in periodicals, or received by direct mail, to gain an understanding of a program's capabilities. If the creator of the program, or those selling it, cannot write an ad without errors, I fear the program (which is certainly more complex than the ad) will be similarly flawed. **SoftSide's** ads are usually error-free but I have seen ads printed elsewhere where, month after month, the errors remained uncorrected.

My experience, while not extensive, suggests that purchased programs with spelling errors in their advertisements, introductions, or instructions (in the manuals or programs themselves) are deficient in other ways as well. Of course, there's no assurance that a well-written, properly spelled advertisement means that a program is flawless or that it accomplishes all that it claims.

Bruce E. Baker
Jericho, NY

PIRACY

Dear **SoftSide**,

I'm in agreement with Scott Adams' reply to Eli Passin that to give away software hurts the hobby (July issue.) If anyone copies a program that he has purchased, for backup purposes, he obviously thinks highly of that program. He feels it was worth the price and he doesn't want to lose it if his computer decides "eat it." If it's not worth the price, why waste media space to make a copy?

Mr. Passin feels it is fine to give programs to friends. My question to Mr. Passin would be, "How many of your own programs, written by you, do you give away?"

What is a give-away program worth to the person receiving it? More than likely, the amount paid — NOTHING. Showing a person a program you feel is excellent should have one of two reactions. Either he is impressed and feels the purchase price is a good value or he feels you have wasted money and good disk space. In the first case, you have given your friend something to look forward to — buying an excellent piece of software. In the latter case, why would they want to see the program again, least of all have a copy?

I feel everyone at our local user's group to be a friend of mine. Now, using Mr. Passin's theory of piracy and consumer's rights, I have just purchased a program that I feel is the greatest thing since sliced bread. It's so good, I want all my friends to have a copy of it. There are about 100 members in our user's group but only half show up at any given meeting. I go down to the local computer store and buy five boxes of diskettes. I spend a night making fifty copies of this great program.

I am not a wealthy man, so I feel I should be reimbursed for the money I put out and the energy (personal and electrical) that I have expended. So I start to figure...\$32 a box for the disks plus 4% sales tax...about four hours time to make fifty copies. I decide to charge my friends \$5 each for what is now an excellent program, divided (or cheapened) by a factor of fifty. According to Mr. Passin, I am a pirate but I also fit into his definition of proper usage!

To all software pirates: when you find an excellent program, why not learn from it and write your own version? You can sell it, give it away, throw it away, or burn it! I feel this is a fair solution but I also believe many pirates are far too lazy to do this.

If someone is attending a group or club meeting, and someone tries to pull a piracy act, just call the police. If you're worried about the dime, I'm sure the author of the pirated program would be overjoyed to pay for the call!

Dennis S. Lewandowski
Dearborn, MI

Dear **SoftSide**,

In reference to the June issue (page 47), Ed Ting had a good idea which I've used extensively since then. I have found that even the cheapest tapes (\$.65 for 3: cycle 60:Sears) have plastic tabs that slide out to release the tape from the center reel. These tabs snap back in to secure the new tape. No messy splicing is necessary.

Be sure to tape your program first — then transfer it to the new cassette. That way, you get only the tape you need. I use both BASF and TDK normal bias. Both work quite well and my two-year old loves to play with the cheap tape that gets thrown away.

Charles T. Vono
Beale AFB, CA

Dear **SoftSide**,

I've been a subscriber to **SoftSide** on cassette since December of 1980. It has been an experience of monthly frustration and anger — in that order.

Upon receipt of your initial tape, I found I was unable to load it in my Model I, 48K, TRS-80 computer. Assuming the worst, I returned the defective tape for a replacement. The replacement was a duplicate — no go.

As the months passed, I noted that all tapes I received refused to load without going through a volume-sensitive LOAD procedure for each program segment. None responded to the prompt "PROGRAM 1, 2, 3, etc.."

R. E. Noble
Prince George, VA

Editor's Reply: We are acutely aware of the cassette loading problems experienced by S-80 owners and we are experimenting with several methods of solution. The S-80 is notorious for loading problems with dropped bits; and misread data is all too common. The Pocket Tape Dubber (**SoftSide**, November 1980, page 85) may help.

DATA BASE

Dear **SoftSide**,

I enjoy **SoftSide** very much and want it to keep coming. The new format is great! Keep up the good work.

I would like to make one request. Would it be possible for you to publish the complete text of "DATA BASE" that appeared in several issues? I now have it working for me with the exception of the multiple-search routine. I would very much like to see the complete and final text.

Again, thanks for a great computer/hobbyist magazine.

Robert Gore
Bloomington, IL

Editor's Reply: You have your wish. See the complete Apple version in this issue with Atari to follow next month.

ROSES AND THORNS

Dear **SoftSide**,

You have to be out of your minds! You are going to sell me six adventures, on diskette, for \$45. That is \$7.50 per diskette. The disk alone is worth \$3, so you are selling the program for \$4.50. You have to be crazy to sell an adventure program for that price. However, I have never been one to pity, so here is my \$45. If you are going to sell for that price, I'll just take advantage and buy it.

Seriously, this is far better than publishing them in the magazine. After you've typed in an adventure, it loses much of its pleasure.

James Hogue
Levelland, TX

Dear **SoftSide**,

Since you seem to ask readers for their comments, I feel the need to make one of my own. I don't like magazines —with one exception— **SoftSide**. It has been a year since I first purchased my Apple and took out a subscription to **SoftSide**. During that time, I have watched, with interest, the process of evolution that has been going on. I like it. It seems there has been a continuous effort to improve the quality of the

continued on page 8

SoftSide™

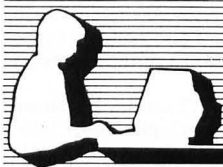
INPUT POLICY

SoftSide Magazine welcomes your comments and thoughts on both the magazine and the field of microcomputing. We try to publish as many of our readers' letters each issue as we can.

For the sake of clarity and legibility, all letters should be typewritten and double-spaced. Send your letters to:

SoftSide Publications,
Input
6 South St.,
Milford, N.H. 03055

We reserve the right to edit any letters prior to publication.



Hints & Enhancements

TAKING THE RANDOM OUT OF S-80 RND

Have you ever tried debugging a program that uses the RND function? If so, you know how hard it can be since the random numbers will be different every time the program is run. This problem can be easily eliminated.

S-80 BASIC maintains a three-byte seed number in RAM which is used to calculate random numbers. By placing constant values into these locations at the start of a program, you will get the same sequence of random numbers every time it is run. The following is a sample program using this technique:

```
10 POKE 16554,1:POKE
16555,2:POKE 16556,3
20 CLS
30 FOR T=1 TO 40
40 PRINT @RND (960), "***";
50 NEXT T
90 GOTO 90
```

Every time this program is run, it will produce the same pattern, despite the use of the RND function in LINE 40. If you delete LINE 10, the program will produce a different pattern each time it is run.

Rich Bouchard
Amherst, NH

SOUND FOR TRS-80

In the June issue, Shane Causer suggested that TRS-80 users could have sound by using a PRINT#-1, " command. This outputs 255 bytes of binary zeros to the cassette player (or your audio amplifier.) This is the same leader that precedes a program saved on tape.

It is possible to get different sounds by printing something other than a null string (""). One method is to send out a string in a literal (defined within quotes

```
PRINT # -1, "XXXXXXXXXXXXXXXXX-
XXXXX
XX").
```

This involves a lot of typing. A better method is to use the STRING\$ command. As an example, use:

```
PRINT # -1, STRING$(255,255), or
PRINT # -1, STRING$(255,0)
```

The first one will generate the 255 bytes of zeros (there is no way to get rid of

these) and then 255 bytes of ones. This results in the purest tone available through this method. The second method will generate a total of 510 bytes of zeros being sent out.

To understand why this happens, remember that the TRS-80 outputs data in a serial fashion — in a string of ones and zeros. All bits sent out have a start pulse. These come two milliseconds apart. "One" bits have a second pulse, one millisecond after the start pulse. A zero bit has no second pulse. When many ones are strung together (as when outputting a "255"), you get a smooth series of pulses (start pulse/"one"pulse, etc.) When ones and zeros are mixed, the series of pulses is not as smooth and the tone sounds raspier.

Note: see pages 46 and 47 of the "Technical Reference Handbook" for a more detailed explanation of how the cassette port works. This includes a graph of the waveform of the output signal.

Geoff Dunn
Lenox, MA

BOWLING

There is a slight problem with the Apple "Bowling" program (November 1980.) No sound occurs for the first ball of the first frame. This is fixed by moving LINE 311 to LINE 7.

Sound can be added to "Collision" (November 1980) with the following changes/additions:

```
1014 SC = 0 : RK = 0 : PRINT
CHR$(7);CHR$(7);CHR$(7)
225 SOUND = PEEK(-16336)
-PEEK(-16336)
```

Sound can be added to "Dogfight" (June 1980) with this addition:

```
3015 SS = PEEK(-16336)
-PEEK(16336)
```

Adding the following line to "Space War" (Apple Edition, April 1980) will cause the opponent's energy level to go to zero when it is hit.

```
605 W(NOT I) = 0
```

The Applesoft music routine in the article, "Programming Hints," (Apple Edition, June 1980, page 65) doesn't work because "VAL" is a reserved word. Use "VL" instead.

Michael Ching
Honolulu, HI

SLIGHT MODIFICATIONS TO FORMAT DECIMALS

The FORMAT DECIMALS routine (Apple Edition, March 1980) does not quite work correctly due to rounding-off problems. For example, LINE 380 doesn't work right with certain numbers — ie. 2312.2. To fix this problem, LINE 380 should be changed to:

```
380 IF 10*N-INT(10*(N+1E-5))
× 1E-5 THEN NN$ = N$ + Z1$
```

With LINE 380 corrected, there is no need for LINE 390 and it can be deleted.

There is also a problem with aligning numbers with zero values. Splitting up LINE 320 seems to be the solution.

```
320 N$ = STR$(N)
390 NL = LEN(N)
```

With the new value of NL, the tabbing becomes misaligned. LINE 400 should be changed to:

```
400 PRINT VAL(A$(I));HTAB
(30-NL):PRINTN$:NEXT
```

While I'm at it, there is little need for using the variables Z1\$ and Z2\$. Their values can be entered directly:

```
360 IF INT(N) = N THEN N$ =
N$ + ".0"
380 IF 10 * N -INT (10*(N + 1E - 5))
< 1E - 5 THEN N$ = N$ + "0"
```

This shortens LINE 100:

```
100 TEXT : HOME : DIM A$(100):T
= 1
```

As it stands now, entries with values less than one do not have a zero printed to the left of the decimal point (ie., .50 instead of 0.50.) By inserting the following line, this will be taken care of:

```
385 IF VAL(N$) < 1 AND VAL(N$)
> 0 THEN N$ = "0" + N$
```

A better style is achieved in LINE 240 with the use of a semi-colon instead of a comma:

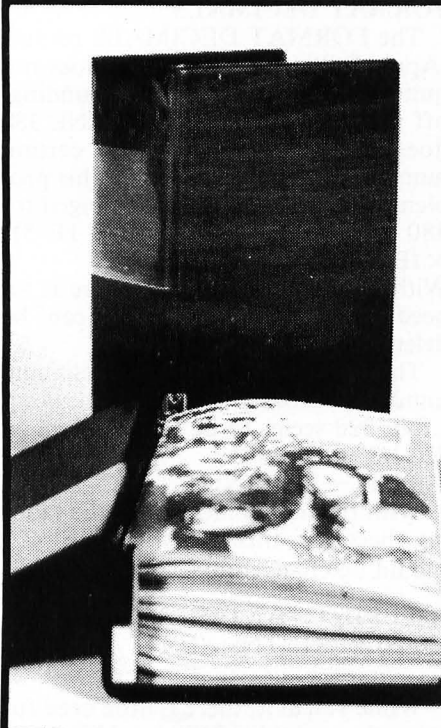
```
240 HOME : PRINT "YOUR EN-
TRIES"; : HTAB 20 : PRINT
"RIGHT JUSTIFIED" : PRINT
```

Sorry, but try as I might, I could not find a single thing wrong with LINES 140, 160, 260 or 280.

Michael Ching
Honolulu, HI

continued on page 9

Protect Your Investment!



With SoftSide™ Vinyl Binders

Collectors! Protect your SoftSide back issues, Volumes I and II, or any publication of your choice, with these durable wood-grain vinyl binders with inside pocket and clear spine sleeve for easy identification. Holds and protects 12 back issues. A regular \$4.95 value. SALE priced at \$3.95*. FREE (while supply lasts) with the purchase of Volume I or II (12 issue collection of SoftSide).

SMALL \$3.95
8½ x 11 \$7.95

SoftSide™
6 South Street Millford NH 03055

Input

continued from page 6

magazine and the courage to experiment. Considering the current cost of a flop, this is nice to see. Most magazines pick a format and stick with it; changes are few and far between. Most would hold to the philosophy that dull and certain is better than exciting and uncertain. It is nice to see you make the effort.

Please continue to include programs and material other than games. While I know that the entertainment content of your magazine is well received, the only way I get to read SoftSide before my sons is to get it first and not let it go. I find that what I go back to is "DATA BASE", three-dimensional figures, sound routines and other non-game programs.

Keep up the good work and continue to make the changes you feel are necessary, even at the risk of causing some confusion. The benefits are far more obvious than the defects.

David Marlowski
Pago Pago, American Samoa

Dear SoftSide,

Printed in the July (page 10) and August issues (page 22) is the following: "We will allow three months after initial publication of a program for the translation to be sent, and the winning entry will be published the fourth month." I was getting ready to send in my translation of "Dairy Farming" (Atari version) until I bought the August issue. Looking at the Table of Contents, I found the translation was already in for that program. What's the reason for this? Three months hadn't passed yet.

There are probably others out there in Atari Land and Apple Land who also got cheated out of a chance at that software certificate. I do think SoftSide is a great magazine but, when you print one thing and do another, that just isn't right.

Unlike some other Atari users, I won't complain about the dominance of S-80 material because I use one in school. If you don't mind the daily lock-ups, trips to the repair shop, and other nuisances, it's a good computer.

How about using better quality staples? Almost every issue of SoftSide I own has either no cover or the cover is falling off. The center pages also fall out if used too much.

Message to all Atari and Apple users: if you are interested in joining a user's group in the Albany, Schenectady or Troy area, please call "The Computer Room" at (518)869-3818

P. J. Maloney
Schenectady, NY

Editor's Reply: Good points! We have changed the text of the translation appeal to clarify some of the misunderstandings. You may have noticed we began using three staples in the binding of the September issue. Our printer has assured us this should help the durability of SoftSide.

Dear SoftSide,

I'm glad to see that you published my program, "Battlefield." I wasn't too happy to see the way you removed my REMarks to replace them with the exact same thing in external com-

ments; nor the way you reduced the board, for no comprehensible reason. I also noticed changes in the documentation but most of those were improvements.

There is one change I can't accept. If you will recall, I very cleverly put each logical block into a separate subroutine, then used a main program consisting of a series of labelled GOSUBs, and included a basic flowchart as part of the program. Did you leave this alone? No. Did you delete the GOSUBs and corresponding RETURNS (which would save a few bytes and tenths of a second) at the cost of considerable clarity (as the removal of each subroutine's REMarks presumably accomplished?) No. Instead, you deleted the REMarks at the end of each GOSUB line. This destroyed the whole point of the procedure, without deleting the GOSUBs themselves.

In closing, I would like to add that I was flattered that you thought enough of "Battlefield" to translate it to Apple and Atari. Thank you.

Joe Humphrey
Topeka, KS

Editor's Reply: I can't defend the editing we did on "Battlefield" as the only way to present a program listing. But we do reserve the right to edit programs and we try to be reasonably consistent in the way we document program listings. Part of that consistency involves deleting REMs and splicing in mixed upper and lower-case comments. We feel this makes the documentation stand out better in the magazine listing. It's up to individuals, then, to add REMs to their programs if they so desire. We freely admit this is an arbitrary format, but at least we're consistently arbitrary!

Anyway, since you feel so strongly about it, we're happy to publish a listing of your original REMarked lines.

List 80, 160
80 GOSUB 1000: REM GIVE INSTRS
90 GOSUB 2000: REM INIT VARS
100 GOSUB 3000: REM GET NAMES
110 GOSUB 4000: REM DRAW BOARD
120 GOSUB 5000: REM BEG W/ BL
130 GOSUB 6000: REM MAKE MOVES
140 GOSUB 7000: REM DO BATTLES
150 GOSUB 8000: REM END TURN
160 HOME : END

Dear SoftSide,

I have been a subscriber to SoftSide since early this year. I was beginning to agree with all the letters in your magazine which complain about your program listings having so many problems. However, when my June issue arrived, I keyed in all the programs and they ran terrifically.

My husband and I particularly liked the "Analist" program as it truly makes following the flow of a lengthy program much easier. I also thought it an excellent idea to publish "Old Glory" and "Word Search Puzzle" for the S-80, Apple and Atari. It would be nice to have one or two listings like that in each issue (especially for those of us who cannot convert one computer listing to another).

I would like to know if you have published (or plan to publish) any articles which will explore the differences in the S-80, Apple and Atari languages, going into detail about the process of converting one language to another. I would very much like to learn the comparisons of all three.

Rochelle McNutt
Lynnwood, WA

Editor's Reply: A series of articles comparing the systems we support is in the works. Watch for it in an upcoming issue.



Hints & Enhancements

continued from page 7

IMPROVE "INTERNATIONAL BRIDGE CONTRACTORS"

In your April 1981 issue, I input the program, "International Bridge Contractors," and found that it ran a bit slow. It took between 250 to 300 moves to finally reach Chairman of the Board.

As you will note in the listing, I have made some changes based on my experience as Traffic Manager/Customs Broker and Foreign Freight Forwarder. I found these changes made the game more exciting.

In an average game with four players, by at least the 25th turn two players would have gone bankrupt; one player would still be Office Manager; and the last player would have reached the position of Regional Supervisor. Please note the randomness of events is unaffected as I increased the random factor from 40 to 50 and changed the conditional statement which prints "NOTHING OF IMPORTANCE IN THE NEWS" if the random number generated is less than 3 or greater than 40.

Try this for yourselves with a copy of the program, making the necessary adjustments.

```
500 RESTORE:CLS:XRND(50):PRINT"YOUR SECRETARY REPORTS:";P
RINT@:IFX(30R):40THENPRINT"NOTHING OF IMPORTANCE IS IN TH
E NEWS.":PRINT@:PRINT;GOSUB2000:RETURN
503 DATA"CONGRESS HAS JUST PASSED A HIGHLY RESTRICTIVE
TARIFF ON IMPORTED STEEL WHICH COSTS THE COMPANY
$5,000,000.",5000000
504 DATA"COMPANY LOSSES TOTAL $4,000,000 BECAUSE OF
LATE DELIVERY OF CONSTRUCTION MATERIALS FROM A
THIRD WORLD COUNTRY.",4000000
505 DATA"A SUSPENSION BRIDGE CONSTRUCTED BY THE COMPANY
FIVE YEARS AGO COLLAPSES DUE TO FAULTY ENGINEERING,
WITH LAWSUITS TOTALING $10,000,000.",10000000
506 DATA"COST OVERRUNS COST THE COMPANY $3,500,000 BECAUSE
OF LATE DELIVERY OF CONSTRUCTION MATERIALS TO
JOB SITE BY AN 'EL CHEAPO' TRUCKER.",3500000
507 DATA"COMPANY IS PROSECUTED BY THE JUSTICE DEPARTMENT
FOR ILLEGAL PAYOFFS TO OVERSEAS SUPPLIERS.
FINES TOTAL $6,500,000.",6500000
508 DATA"PREGNANT FORMER SECRETARY SEEKS $10,000,000
FOR 'ILLEGAL ENTRY'. COMPANY SETTLES OUT OF
COURT FOR $500,000.",500000
509 DATA"CHARTERED VESSEL CARRYING $5,000,000 IN
STRUCTURAL STEEL FORMS SINKS MYSTERIOUSLY
IN MID-OCEAN AND UNDERWRITERS WILL NOT
HONOR CLAIM.",5000000
510 DATA"U.S.CUSTOMS FINES COMPANY $10,000,000
FOR FRAUDULENT INVOICING ON IMPORT SHIPMENTS
DURING THE PAST FIVE YEARS.",10000000
526 DATA"A FORMER CLIENT HAS GIVEN THE COMPANY
$20,000,000 BECAUSE OF ITS PERFORMANCE ON THE
JOB.",20000000
527 DATA"NEW LABOR CONTRACT ACTUALLY SAVES THE
COMPANY $5,000,000.",5000000
528 DATA"COMPANY CAPTURES CORPORATE SPY AND
RECEIVES $5,000,000 IN REWARDS FROM APPRECIATIVE
COMPANIES.",5000000
529 DATA"COMPANY MAKES EQUIPMENT PURCHASE ON
FOREIGN MARKET AND SAVES $20,000,000.",20000000
```

```
530 DATA"COMPANY RESEARCH DISCOVERS NEW
COMMUNICATIONS DEVICE WHICH SELLS FOR
$30,000,000.",30000000
531 DATA"ACCOUNTING DEPARTMENT REPORTS PREVIOUS
EARNINGS UNDERSTATED, COMPANY GAINS $3,000,000.",3000000
532 DATA"GOVERNMENT LOBBY FORCES LEGISLATION
THROUGH CONGRESS WHICH SAVES THE COMPANY $10,000,000.",100
00000
533 DATA"STOCKS WHICH THE COMPANY OWNS EARN
DIVIDENDS OF $40,000,000.",40000000
534 DATA"NEW ACCOUNTING SYSTEM SAVES COMPANY
$4,000,000.",4000000
535 DATA"MAJOR COMPETITOR GOES BANKRUPT.
COMPANY GAINS $50,000,000 IN NEW REVENUES.",50000000
536 DATA"COMPANY INSTALLS NEW COMPUTER SYSTEM
AND SAVES $5,000,000 IN LABOR.",50000000
```

John T. Hyatt
New Orleans, LA

MORE MODIFICATIONS

Harland Hill's Adaptation of George Blank's One-Liner (July 1981, page 32) can be even more useful with a few modifications. The length of the string output can be increased to 220 characters and displayed until you hit BREAK (or the next power failure occurs.)

I found that doubling the time delay made it easier for people to read, but the loop in LINE 140 can be adjusted to any speed. I also noted that it was a pain in the neck to count how many characters I had with a long string. Therefore I added the "X—STOP HERE" display. A message of any length may be input until the cursor reaches that point. The usual admonitions regarding punctuation apply in this program.

Karl H. Meyer
Lake Worth FL

REWARD!

TRANSLATION APPEAL

SoftSide will give a \$100 software certificate to the author of the best translation of a past SoftSide feature program. Each month we will publish at least one of these translations. Your portfolio will be enhanced to say nothing of your software library!

We will accept entries for all past SoftSide programs at any time. However, we suggest you submit translations of recent programs within three months of their original publication date for maximum consideration. Entries must be submitted on tape or disk, accompanied by complete documentation. Please enclose a self-addressed stamped envelope if you would like your entry returned.

The quality of each translation will be judged by the SoftSide editorial staff and prizes will be awarded at the time of publication.

ATTENTION AUTHORS

SoftSide Publications is actively seeking programs, article and review submissions for the TRS-80™, Apple and Atari home computers. This is a chance for programmers as well as users to make some money to help pay for the "computer addiction" and get their efforts out where they can be appreciated.

Programs — SoftSide has always been the leader in the field of BASIC software and BASIC remains our specialty. However, with the advent of Disk Version (DV), we can now also offer an outlet for Machine Language and multiple language programs which do not lend themselves to printed versions. Games, utilities and educational software, as well as any other applications for the home computer user are preferred, although we will consider virtually any type of program. Hybrid mixes of articles and programs are also welcomed.

When submitting a program, please be sure to include full documentation of subroutines and a list of variables, as well as a brief article describing the program.

Reviews — Well written, informed reviews of all software for the systems we cover are a regular feature of SoftSide. Reviewers should take into consideration all aspects of a particular software package, from speed of execution to programming creativity to the estimated length of time that the product will hold the customer's interest.

Articles — We welcome article submissions of all types, but prefer those specifically geared to the home computer market. We give our readers information as a first priority, but vary our content to include some humor and commentary.

All text, including documentation and descriptive articles for programs should be typewritten and double-spaced. Extra monetary consideration will be given to articles and reviews submitted on machine-readable media (Scipsit, Super-Text II, etc.). Programs should be submitted on a good cassette or disk. TRS-80™ BASIC programs should function under both Level II and Disk BASIC.

Send to:

SoftSide Publications
SUBMISSIONS DEPARTMENT
6 South Street
Milford, NH 03055

We regret that due to the volume we receive, we are unable to return submissions.

Be sure to send for our **Free Author's Guide**. It further outlines the specifics of our submission procedure.

TRS-80 is a registered trademark of Tandy corporation.



Outgoing Mail

by Randal L. Kottwitz

Salutations! The excitement I spoke of brewing in the halls of **SoftSide** last month is bubbling over the sides in this one. With the release of the major project, I-String, a whole new vocabulary is being spoken around here. As you, the readers, need to understand what we print in these pages, the time has come for a few new definitions.

I-String — The concept of "Intelligent String," utilizing the technique of storing data in a string identified by a character. In other words, I-String is an idea, not a program.

Envyrn™ — A problem-solving utility, based on the I-String concept. In essence, Envyrn™ is a database manager/editor whose prime data for manipulation are graphic symbols rather than numbers or words.

Envyrnment™ — A database created through the use of Envyrn™. Envyrnments™ will contain single subjects (Star Trek, Flight Over the Himalayas, United States Map, etc.) and be dependent on an included interpreter. In order to carry the label, "Envyrnment™," a database must be reviewed and approved by **SoftSide** Publications.

Diversions thru Envyrn™ — A subscription magazine/media combination containing Envyrnments™ and accompanying background information. Subscribers will receive six Envyrnments™ per year with comprehensive booklets, containing extensive maps, history and any other appropriate information pertaining to the Envyrnments™. The game playing interpreter will be included with initial subscription to **Diversions™** with further modifications to be included with future Envyrnments™ as necessary.

SoftSide DV — Another mixed media presentation consisting of the programs published in **SoftSide**, the magazine, and additional programs not published in printed form due to their length, complexity or computer language used. DV is the "deluxe" version of **SoftSide**, not its replacement.

SoftSide on Cassette — A cassette tape version of the programs published in **SoftSide**, the magazine. At this time, the tapes contain only programs from **SoftSide**. Enhancement of the tape version is under consideration, but no decision has been reached on this matter.

SoftSide — The mother to all of the above! **SoftSide** will continue to publish the highest quality software available and act as the printed accompaniment to DV and the cassette version. The latest news concerning Envyrn™ and **Diversions thru Envyrn™** will also be published in **SoftSide**.

These definitions should help you to better understand what is under way at **SoftSide** Publications.

In response to the many complaints we've had concerning the way your **SoftSides** are wearing in the mails and in use for typing in programs, our printer has agreed to insert a third staple in the binding. Please let us know if it's helping.

By the time you read this, the fall computer show circuit will be well under way. If you've not already done so, please drop by and see us. We value your input greatly, and there's no better opportunity than face-to-face at the shows. I'm planning to attend as many of them as possible and will report any remarkable developments in this column in the upcoming months. ☺



About This Issue

SoftSide is three years old this month and do we have a birthday party issue for you! Surprises abound from cover to cover.

☐ **I-String Produces Envyrn™** is our cover feature. The words to describe the concept behind Envyrn™ haven't yet been written. We've been pulling our hair out, trying to find the best way to present this idea. We'd suggest you begin with the definitions in **Outgoing Mail** and work your way through to the centerfold/poster. (Who'd have ever thought **SoftSide** would have a centerfold?) This is literally only the tip of the iceberg for **Envyrn™**. After reading the material in this and upcoming issues, you'll agree that "entertainment software will never be the same!"

☐ As promised, **Developing Database** has returned! Mark Pelczarski has added some new features and solved some

old problems in the Apple version and we are happy to print the complete listing. Watch for the complete modified listing of the Atari version in an upcoming issue!

☐ **Leyte**, a simulation based on the battle of Leyte Gulf, brings the great names of World War II to the screen of your micro. See if you can better the decisions of Roosevelt and MacArthur. Victor Vernon has supplied the original S-80 version, with Apple and Atari translations by Alan J. Zett.

☐ Most Atari owners have heard that they can create their own character sets, but few have had the intestinal fortitude to give it a shot. Now, Alan Zett has simplified the process with **Character Generator**. He's then given us a practical application in **Take Apart: Atari Quest**. Get out those August issues and try this enhancement.

☐ All of you S-80 owners can start spending those quarters you've been saving for the arcade on something else. **ABM Command**, by Arnold E. van Beverhoudt, should keep you glued to the keyboard for many, many hours.

☐ Plus, **Plotting the Yang/Yin Symbol** by David Delli Quadri, **One-Liners**, **K-Byters**, **The Sensuous Programmer** and much, much more.

Next month, we'll present a special theme issue, **Computer Music**. Featured will be reviews of the major hardware advances in music, software to make your micro a musical instrument, Rimsky Korsakoff's **Flight of the Bumblebee** by Morris and Cope and many other special features related to this rapidly expanding field. Until then, happy hacking! ☺

CALENDAR

October 2-4

Microcomputers in Education College Park, MD

A series of workshops is being offered to Technical Education Research Centers (TERC) designed to meet the growing needs of educators in the elementary through college levels. This will be a hands-on experience and will cover educational issues as well as microcomputer languages and applications. Extensive workshop reference materials will be given to participants and hotel accommodations will be available.

Contact: TERC, 8 Eliot Street, Cambridge, MA 02138, (617)547-3890

October 5-6

National Conference on Software Shoreham Hotel, Washington, DC

Prominent DP professionals will cover nearly a dozen topics during this two-day meeting sponsored by The Commission on Software Issues in the 80's. Featured speakers will include the noted author and consultant, Daniel D. McCracken, who will present the keynote address; Martin A. Goetz, Senior VP of Applied Data Research, will be the second-day speaker; and Rep. George Brown (D-Calif) will deliver the luncheon address, discussing his bill, "The Information Science and Technology Act." Sessions will be open for discussion following each report and will be led by Daniel T. Brooks, Commission/Conference Chairman. Contact: Conference Manager, U.S. Professional Development Institute, 12611 Davan Drive, Silver Spring, MD 20904, (301)622-0066

October 12-15

Information Management Exposition & Conference/Info '81

Coliseum, New York, NY

Prepackaged software, both customized prepackaged and custom-designed will be discussed. In addition, hardware and software exhibitions will be displayed. Contact: Clapp and Poliak Inc., 245 Park Avenue, NY, NY 10167, (212)661-8410

October 17

Educational Computer Fair Cuyahoga Community College, Cleveland, OH

A one-day educational fair, produced by and for educators in the elementary and secondary grades, will be presented to the novice through experienced computer user. Over 20 workshops are planned with curriculum areas covering language arts, math, social studies, science, business, and computer literacy. Other topics will include computer math and applications, using computers in the elementary grades,

Computer Managed Instruction (CMI), selecting a computer and available computer grants. A hands-on approach will be an enhancement to this educational fair. Hardware and software exhibits will be displayed throughout the day by various vendors. Comparisons of TRS-80, Apple and Atari computers will be held. Advance registration (including lunch) is \$10.

Contact: Ellen Richman, ECC Coordinator, 245 Meadowood Lane, Moreland Hills, OH 44022, (216)292-4655

October 20-22

Computerized Office Equipment, Expo Southwest

Astros Hall, Houston, TX

Exhibitions will include office equipment and word-processing systems. Approximately 100 exhibitors are slated to participate in this expo on office automation.

Contact: Industrial Scientific and Conference Management Inc., 222 W. Adams Street, Chicago, IL 60606, (312)263-4866

October 19-23

Systems 81

Munich, West Germany

Applications and computer systems will be focused on in this four-day conference.

Contact: Kallman Associates, 30 Journal Square, Jersey City, NJ 07306, (201)653-3304

October 27-29

Computer Graphics '81

Regent Centre Hotel, London, England

An equipment presentation will be on display. Sessions are planned around graphics systems hardware and software, animation, image processing, business and home graphics.

Contact: Online Conferences Ltd., Argyle House, Northwood Hills, Middlesex HA6 1TS, England

October - December

Seminars from Management Information Corp (MIC)

Seminars are planned to be held at various locations throughout the United States. These are designed to fill the needs of business people needing an introduction to the selection of systems and their usage. Complete schedules, fees and locations are available upon request. Contact: Carol Bell, c/o MIC, 140 Barclay Center, Cherry Hill, NJ 08034, (609)428-1020

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.

Nine Games for Preschool Children

by George Blank

Even pre-schoolers deserve a shot at the wonders of microcomputing. With these nine games, they not only will have a chance to tickle the keyboard, but learn letters and numbers to boot. And if that isn't enough, they'll have a good time doing so. What more could a parent ask for? Here are education and entertainment for the very young in a single package!

S-80, Level II, 16K, Cassette \$9.95



SoftSide
Selections
6 South Street Milford NH 03055

Let PASCAL-80 talk some sense into your computer

Phelps Gates, the author of "APL-80", brings you "Pascal-80" for your S-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 S-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.

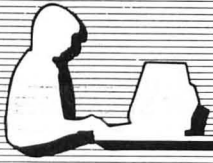
Functions: Abs, arctan, call, chr, cos, eof, eoin, exp, inkey, in, mem, odd, ord, peek, pred, round, sin, signif, sqr, sqrt, succ, and trunc.



"Pascal-80" does not implement variant records, pointer and window variables, or functions and procedures used as parameters.

S-80 32K Disk.....\$99.95





My Side of the Page

GETTING A BIT SERIOUS — PART 5

by Lance Micklus

The first time I mentioned the "Mean Craps Machine" was six months ago. At that time, I pointed out how long it takes for a product to be developed and get to market. (Sometimes it feels like forever.) As of this writing, (July, 1981), the S-80 Color Computer, Model I and Model III versions are completed. The documentation has been written, and one copy of the program has been sold to a guy who logged onto my bulletin board and saw it in my catalog of items. Right now, here's where my work log stands:

1. About 125 hours of work developing the program on the Color Computer and testing it out.

2. Another 50 hours of conversion work on the Model I.

3. About 8 hours getting the documentation ready, including proof-reading and a cover picture.

4. An additional 50 hours to do further testing of both versions, playing and re-playing the game.

If we assume that an hour of work costs a modest \$10 — for labor, use of equipment, insurance, taxes, and general overhead — then the total cost comes out to be $(125 + 50 + 8 + 50)$ hours times \$10, or \$2,330. Assuming a price of \$10 a copy, with a 15% royalty, the number of copies I need to sell to break even is 1,554. Total sales to date: one.

I have sent the program to Scott Adams for evaluation. But I can't assume he is going to accept everything I send him. So, in addition to being out \$2,330, I have no certainty that this program will ever get to market; and as of this moment, I don't have a distributor, either. It's now a game of wait and see.

Let's assume, though, that Scott Adams does like the "Mean Craps Machine." In that case there will be an additional length of time I must wait while he gets the package together and adds the program to his advertising and catalogs. That will probably take another two to three months, which means you'll be seeing it offered for sale sometime just before Christmas. By the time you buy it from Scott and he pays me my first royalty, it will be at least January of 1982.

Excluding whatever sales I make on my own in the meantime, the total length of time, from the point work first started on the program until the first royalty check arrives, should be about ten months. Naturally, the first check will not be for \$2,330. Realistically, it might take a least an additional year to earn my money back. But the "Mean Craps Machine" is just a single 16K BASIC program. Imagine how much **more** money must be invested, for a much longer period of time, for programs like "General Ledger," "ST80-III," "Profile II," and "LDOS." Just the cost of writing documentation for any of these programs is far greater than the total cost of the "Mean Craps Machine." "LDOS" cost \$27,000 — **JUST FOR THE DOCUMENTATION!** The cost of documentation for "ST80-III" was over \$4,000, exclusive of printing costs.

There's one other thing about my program that's worrying me — do I have the rules straight? Nowhere have I been able to find any kind of official rule book. The best I could come up with was the instruction manual for the Craps layout my wife and I bought in Atlantic City. The manual was later found to have errors in it. The official rule book for "Caesar's Boardwalk Regency Hotel" explains the game briefly — just enough so you can play. If you have questions on the finer points, it tells you to ask one of the hotel's dealers. (I wonder if that includes calling from Burlington, Vermont and asking to speak to a Craps dealer so I can ask him a question?)

Fortunately, I hadn't sent the game to Scott before reading the INPUT column in the July *SoftSide*. Son-of-a-gun, if I hadn't forgotten a small rule regarding bets on the odds. Then I discovered a bug in the program. This points out one of the most difficult decisions that a programmer has to make — when do you quit and say it's ready?

Bugs are a problem all programmers must face up to. I found that I just had to accept the fact that no matter how hard I try, nothing will ever be perfect. If I keep trying to make it perfect, in the end I will produce nothing. Fortunately, computer users seem to be

able to forgive mistakes. What bothers them is sloppy work. The goal then, is to create a carefully checked out program, and hope that if something bad comes up it won't be too serious.

There is a real advantage to making up your own game. At least no one can come back to you and challenge you on the official rules. But in games such as Craps, there are fixed rules (unwritten, it seems) and the program must follow those rules to the letter. Although Craps is not difficult to learn how to play, it is a very complicated game. There are many different ways to bet, each paying off under special conditions and according to special odds. People get very upset if these games don't follow all of the rules to the letter; I learned that the hard way from Checkers. But there's one big difference between Checkers and Craps. Most people play Checkers for fun and many don't really know all of the official rules. Minor rule violations will go unnoticed by most players unless they're serious about the game. On the other hand, Craps players usually play for money — big money. They're all serious about the game. The only thing the programmer can do is treat these things as bugs and hope the public will understand.

This month, I want to devote most of my attention to contracts. Simply stated, a contract is an agreement between two or more parties. The agreement basically states the terms and conditions under which things of value are to be exchanged. In law, anything worth a dollar or more is considered a thing of value.

Never enter into a contract with the idea that it will force something to happen — you can't make somebody do something he doesn't want to do — or assume it can prevent you from being cheated. All contracts assume good faith on the part of all parties concerned. If someone wants to cheat you, he will, no matter what it says on paper.

That's an important point. People tend to think that contracts are designed to protect you from getting cheated. The real purpose of the contract is simply to put down on paper the terms to which two or more people

continued on next page

continued from previous page

agree. Thus if a question arises later, everyone has a fixed point of reference. A simple example: Did the XYZ Corporation agree to supply the ABC Company with a source code listing, or a source code disk? It's easy to believe that when you agreed to sell the ABC Company the marketing rights to the program three months ago, you thought you agreed to a listing; but what if they thought they agreed to the actual source code on disk? Details like this can be forgotten in three months, but can be a major source of friction unless they're written down on paper.

In general, when dealing with other businesses, if there is any chance of a misunderstanding, then write down exactly what you have each agreed to. This is not for the purpose of taking someone to court, but to avoid a dispute later on that could harm your relationship.

Some software houses require that the author sign a contract before they will market his programs. Some contracts used in the industry are complicated and raise a lot of good questions and are ideal for discussion purposes.

One golden rule: Never sign a contract unless you understand exactly what it says. What you **think** it says doesn't count. A cover letter that sometimes comes with an agreement is a perfect example. It states that the purpose of the enclosed agreement is simply to indicate that the program you submitted is your own original code. No one could object to a clause like this being in an agreement for software. Sure enough, one item in the agreement does say that. But it is just one of many things in the agreement. When you sign on the line, you're agreeing to everything, even though you might not like some of the other things. So, don't fall into that trap.

Another popular gimmick is to state that the agreement was prepared by a lawyer to be sure it is a legal agreement. There's no law that says a contract has to be prepared by a lawyer. The fact that it has been in no way means that the agreement will really serve your best interests. It is more likely that it means that the publisher has made sure it will serve HIS best interests and will be enforceable in a court of law.

What would we really be agreeing to if we signed a contract that contained the following clause? "Programmer hereby conveys to the Publisher the exclusive right to copy, distribute, list, sell, advertise for sale, prepare derivative works and otherwise utilize said program." Now we get to the part

mentioned in the cover letter. It goes on to say, "Programmer warrants that he has good title to said program and that said program is completely his original work, and that no other person, firm, or corporation has any interest, right or claim to said program, or any part thereof." Well, I don't have any trouble with that part.

It further states, "All rights transferred hereunder are exclusive. Programmer covenants that he has not and will not transfer, assign, release or convey any rights in or to said program or any derivative work based on same to any other person, firm or corporation, without the consent to Publisher." Wow!!!

What all of this really means is this: If I sign on the line, all rights I have to my program become the rights of this publisher, leaving me with none. Not only that, these rights are exclusive forever. In effect, this publisher would have the same ownership rights as if he wrote the program himself. Unless I get some money out of this deal, I am left with nothing — just as if I had never written the program in the first place.

Selling all the exclusive rights to a program for a lifetime is not, in itself, bad. There are times when companies do sell such rights. Suppose tomorrow I decide that I'm tired of writing programs and want to close the door and sell out. I might very well sell my programs under these conditions. Legitimate buyers for a program like "ST80-III" might be Small Business Systems Group, The Microperipheral Corporation (manufacturers of the MicroConnection modems), Emtrol (manufacturers of the Lynx Modems), or Radio Shack. Naturally, I would expect to get top dollar for the sale — especially if they wanted exclusive lifetime rights. In effect, they would really be purchasing the copyright itself. Since the key phrase here is "top dollar", let's look at the money angle on this.

A portion of the contract may read, "The Publisher agrees to pay Programmer a royalty of 20% of the gross receipts from the sale of said program. Said gross receipts shall be calculated and said royalties shall be paid to Programmer, at least once a year, at such intervals, either quarterly or otherwise, as Publisher, in its sole discretion, shall determine." There are good and bad things in this clause.

The 20% royalty is based on gross receipts (whatever the publisher receives, you get 20% of it.) The remaining 80% covers his costs and profit. Keep in mind that this is not always 20% of the retail price. If the publisher

sold your program to a computer store, it is at a discounted price. Your royalty might be 20% of \$5 (the wholesale price) and not 20% of \$10 (the retail price.)

This type of royalty arrangement is still popular today but it is being replaced by two new types of arrangements. The first setup pays a royalty based solely on the **retail price** of the program. It is the percentage that varies. If the sale is direct retail, you get 20%. If it is a wholesale transaction, you get 15% of the **retail price** — not 15% of the gross. This is a better deal for the programmer because, on a wholesale transaction, the older plan would effectively be paying you 10% of the retail price, instead of the 15% you would receive under the new plan.

The second new method is to pay 15% of the retail price, regardless of the publisher's actual transactions. This doesn't sound as good but generally it will pay better than the 20% gross scheme. The reasoning behind this is most of the big publishers sell more goods at wholesale than retail prices. Generally, I've found that for each retail sale there are two wholesale transactions.

The advantage in both of the new methods is they are a lot simpler. Many publishers offer dealers a discount schedule. The more they buy, the less it costs. Thus, your program might be sold at various places to different dealers, making it complicated to keep track of your royalties, if they're based on the gross receipts. Either of the newer plans reduces complexity and errors.

The real problem with certain agreements is the frequency of payments. Some allow the publisher to pay you once a year — more frequently if he's feeling very generous. This means that for a full year, the publisher gets to use your money, interest-free. That's much too long. The argument could be made that some programs don't generate enough royalties to make it worthwhile to pay more frequently. In my opinion, if there aren't enough royalties to make it worth writing a check more often than once a year, the program isn't worth publishing.

Royalties are nearly always paid every month, no matter how small; of course, there are some exceptions. The Source, at one time, paid royalties to me for "Mean Checkers Machine," once every three months — and I think three months should be the maximum amount of time between royalty payments. However The Source did pay royalties monthly if, at the end of

any given month, they exceeded a certain amount. That's a fair arrangement. The Source is presently paying my royalties on a straight basis, and I believe they are doing the same for everyone else, regardless of the amount.

Occasionally I might find a portion of an agreement that is unclear to me. Any time something is unclear, a little red flag should go up. If I were interested in signing an agreement, I would first talk to my attorney and ask him to check it out. Maybe I'm thick and it's O.K.. After all, I'm a computer programmer and not a lawyer.

One agreement says in part, "Programmer hereby agrees to indemnify Publisher and hold Publisher harmless from any and all liability, loss, damage or cost the Publisher may suffer as a result of any claims, demands, costs or judgment arising out of Publisher's unauthorized use of said program, or arising out of Publisher's or Programmer's infringement of any trademark or copyright laws, or other claims of infringement of title to said program." It further states, "Programmer agrees to defend against any claim brought or action filed against Publisher with respect to the subject of the above indemnification, whether such claims or actions are rightfully or wrongfully brought or filed."

I will take this one section at a time. Let's say I submitted a program to Ramworks and they published it. It turns out that the program was really one written by Radio Shack and it's copyrighted. Radio Shack would sue Ramworks, not me, because Ramworks is the one making the tapes; therefore they're the ones who are doing the copying. What a "hold harmless" clause does is say that whatever harm this does to Ramworks must be undone by me. This means repaying Ramworks for whatever damages they had to pay Radio Shack, plus Ramworks' legal expenses.

If that's what the first portion really implies, then it is a perfectly reasonable thing to ask. But being a computer programmer rather than an attorney, I have to wonder enough to ask a few questions. What happens if Ramworks changes the name of my program to a new name, which is someone else's trademark, and then gets sued? If I agreed to the name change, am I liable? Suppose they never asked me about the new name. Am I still liable? These are good questions. They're the type of things you should get absolutely clear before you sign an agreement — even if it does cost you \$20 for legal consultation.

I'm also not sure if I like the second

part. If a lawsuit did result from the use of one of my programs, it's in my best interest to assist the publisher in every way possible. If that's what it means, it states the obvious. However, it does say something about "rightfully or wrongfully brought or filed." I'm not so sure I want to be responsible for a lawsuit wrongfully brought against my publisher as the result of one of my programs. If that's the way it's got to be, then I want to add something that says the publisher is responsible for any lawsuit wrongfully brought against me as a result of his publishing the program. Suppose the disk on which they distribute my program is really made of sandpaper. I'm not going to pay for that damage. The publisher is responsible — let him pay for it.

In general, publishers should take care of the legal responsibilities in all cases **except** where the author is clearly to blame. If program authors have to take on all this added responsibility, then a lot of them are going to back out in a hurry, because now they've got to deduct the cost of legal expenses and liability insurance from their royalty checks. Furthermore, being individuals, they could be held personally liable and could lose their homes, cars, or anything else of value they privately own. In my case, that's not a big problem because I have an attorney, legal expenses, and all kinds of insurance to handle such a crisis should it arise. Also, since I am incorporated, my personal possessions cannot be attached. Legally speaking, I just work here.

After I talk with my attorney, I may find that I'm still not satisfied with part of the agreement. There are two things I can do. First, I might just scratch out those parts of the contract I don't like. If the publisher wants to accept the agreement, excluding the parts I scratched out, that's fine. The other thing I might do is to add a clause, for instance, to make the document read more to my liking.

Another interesting clause found in some publishing contracts states that you must give the publisher six months' written notice to terminate the agreement to publish your program. This is a fair stipulation since the publisher needs time to sell his stock and remove your program from his advertising. The agreements I have with my publishers (written by my attorney) give 90 days' written notice, although we do try to give the publisher more time whenever possible. The exception would be in a situation where I've been cheated. Then I'd demand that the publisher stop all sales immediately. The worst thing you

can say about a publisher is he's dishonest.

My agreements state that I, or a representative of my choosing, has the right to inspect the books during normal business hours. I've never done that, but I have found that when I visit them, as part of the company tour, all the publishers I've done business with show me the books in which they keep track of my royalty payments.

Some contracts are basically for exclusive rights. While the tendency of the software industry is away from exclusive rights, there is a right and a wrong way to enter into these agreements.

Exclusive rights agreements always pay top dollar, and for good reason. With non-exclusive rights, you might be able to make several sales with the same work and make a killing. Since there is no chance of doing that with exclusive rights, you are instead compensated by being paid more than the work is probably worth.

In some fields, you can still make a "multiple sale" even though you're still selling only exclusive rights. Books are often sold this way. There are rights to publish in hardcover or softcover, comic book rights, movie rights, rights to make it a play, and rights to make it a TV show. All are "exclusive" but each one is sold separately.

While there may be a lot of ways to sell exclusive rights, they all have one thing in common — money. Alex Haley would never sell the rights to one of his books strictly on a royalty basis. Instead, he would sell his rights for a fixed fee plus a royalty on top. The royalty is a bonus for a job well done — not the primary source of payment for the rights to his work. This is the way professionals in other industries think. Sadly enough, many publishers in the software industry are just beginning to come around to this way of thinking.

The publisher buys exclusive rights with a royalty — but the royalty should be the bonus. In fact, the publisher might very well end up with ownership rights to the program without ever paying a penny. You, on the other hand, could end up with no program and no money. If someone wanted to buy the exclusive rights to "Mean Craps Machine," I would expect them to pay me double what my expenses were, or around \$5000. In addition to that, I would want a royalty, although it could be lower than the 20% mentioned previously. My message to publishers is simply, "If you can't afford to pay for exclusive rights, don't ask for them."


continued on next page

Another way to negotiate exclusive rights is with a limited-life contract. Basically, the publisher pays money up front plus a royalty, but his rights have a time limit on them — perhaps a year or two. One method I've used is an upfront fee against a royalty. A lot of entertainers use this method in their contracts — you are paid either a fixed fee or a percentage of the gate (like a

royalty,) whichever is greater.

My advice on exclusive rights is to avoid them whenever possible. My experience has been that you're better off without them. You should only consider exclusive rights if the arrangement will pay you more than the product is worth. Historically, however, this industry tends to have the reputation of being a little cheap. The only time you should consider exclusive rights is when the deal gives you upfront money and you need the cash.

Once again we see that all it really boils down to is money.

Publishers claim there isn't much money in this industry — and it may be true. But it seems to me if the publisher is going to insist upon a contract that gives him every possible advantage, and protects him against every possible contingency, he should not expect top-flight authors to stand in line to sign on the dotted line. They cannot afford to take all the risks and just give their hard work away. 

The Publisher's Response to "Getting A Bit Serious, Part 5"

As publishers of **SoftSide Magazine**, we feel a responsibility to present articles of interest such as the one you've just read. Because we publish a monthly magazine, we try to provide assistance and information on the entire spectrum of microcomputing. As a software publisher as well, we would like to respond to the article in order to present what we think is a more balanced view of working with software publishers.

Mr. Micklus has described his experiences for the benefit of those of you readers who are considering pursuing the personal and monetary reward of programming personal software. He has made some fair statements about the frustration and problems which accompany these pursuits.

SoftSide has always encouraged its readers to write their own software. Being limited by a small editorial/programming staff, we rely heavily on submissions from our readers for the content of our pages. When a submission comes along that we believe is marketable by Ramware (our line of software), we'll negotiate a deal with the author. So we're familiar with contracts. And the problems authors have with them. And the problems they present to publishers.

Mr. Micklus has given you a good idea of the things to watch out for if you are contemplating the sale of your programs. And we agree that the more the programmer knows about the "big bad world of business," the better. To the novice, the experience can be intimidating, confusing and frustrating. But please note that we say it **can** be all of those things; it need not be.

Where should you begin to shop around for publishers? It is likely that you follow most of the popular computer magazines. Take a good look at

their ads. Which of these publishers impress you with their advertising approaches? With which have you had the best experiences in ordering software? Which offer selections of software that are compatible with yours?

Approach these publishers first. But realize that, even with the best of them, you're going to have to **SELL** your program — much as you might believe that the program sells itself, publishers aren't likely to be sold just because the program took you 500 hours and several hundred dollars to create. Unfortunately, the heart and soul you poured into your program isn't going to influence the publisher.

Realize too that, as Mr. Micklus points out, software publishing is not yet the wildly lucrative business one might imagine. The publisher puts a lot of money, as well as his reputation, on the line every time he undertakes the duplication, packaging, promotion, and distribution of a new program.

Unfortunately, extensive marketing data on the buying habits of the microcomputer owner have yet to be compiled. So, the software publisher must rely on past sales and on instinct in making many of his decisions. In addition, many publishers are not marketing experts; they are, more often than not, software authors who have found it necessary to become businessmen, in order to exploit their own talent, as well as that of others sharing a common interest.

If the publisher is successful, he has a pretty good idea of what's likely to sell. His business is anticipating the market, knowing how it will respond to a particular program and how best to handle the promotion. From his point of view, you, the programmer, have only completed the first step of the job. It's up to


him to sell it. This means a large investment on his part too.

First, he'll have to pay someone to go over the program with a fine tooth comb to review it, offer suggestions for improvement, examine the documentation — essentially to polish up the product.

Then, there are production costs. Editing and typesetting the manual. Setting up the duplication process. Designing the packaging. Evaluating the best ways to promote the product: press releases, space advertisements, inclusion in a catalog, promotional literature to dealers. As Mr. Micklus points out, neither the publisher nor the author will see a sale for at least two months following the contractual agreement. Chances are, the publisher will spend three or four times what the author has already spent on his program, within the first month of the contract.

No, we aren't taking sides against the software author. We believe that certain obligations, responsibilities and investments on both sides are inherent in every contract. Just as we can't fault the software author for wishing to get the best deal for his program, there is no reason to expect that the publisher ought not to feel the same way.

We know the contract Mr. Micklus refers to. It was chosen as a sample because of its many obvious biases. It is not widely used, however, and is far from being an industry standard.

In closing, our advice is to take care when entering a contract with a publisher, just as you would with any other sort of contract. Don't sell yourself short — but remember, you and the publisher are a partnership. Allow each party to do what he does best and you'll both be better off. And finally, if you've got the energy, ideas, equipment and wherewithal, go to it! 

The Sensuous Programmer

by "J"

The Sixth Euphoric Encounter

Be forewarned that this month's topic is a bit kinky. At least, a lot of people seem to get all tied up in knots when they work with strings. Maybe it's a little optimistic to think that one column will untangle all the ins and outs of working with strings on your computer system, but I'll try to thread my way through anyway.

Last month's discussion of methods to idiot-proof your programs, and previous columns as well, have touched on the use of strings. Let's start back at the beginning with what strings are, and then look at the various ways in which computers can manipulate them.

Strings are simply groups of characters, all strung together in a row and enclosed between quotation marks. Examples of strings are "George Washington", "105586", and "#8-ROY.\$\$!%". Each character in a string is stored in one byte of the computer's memory as a number which ranges from 0 through 255. The capital letter "A" is stored as the number 65, the letter "B" as 66, and so on; the number 0 is stored as 48, the number 1 as 49; a space is stored as 32, an exclamation point as 33, a period as 46. These numbers are known as the "ASCII codes" of the characters; and those which represent common letters, numbers, and punctuation are pretty standard in all computer systems.

The most common way of using strings in programs is to "assign" them to variables, just as you assign number values to variables such as A, B, or X1. Variables which are to be assigned string values must have a dollar sign tacked onto the end, or (optionally, on the S-80) defined as string variables using the DEFSTR statement. Assigning values can be accomplished not only through the use of the "=" sign, but also through statements such as INPUT and GET (discussed in the June issue) and READ (to be discussed in a future column). All of the following examples demonstrate the assignment of a string value to a variable:

```
10 Z$ = "YOUR SCORE IS: "
```

```
300 INPUT NAMES$
```

```
90 READ L$
```



```
(Apple:)
410 GET X$
```

```
(S-80:)
50 DEFSTR RP
60 RP = "RIGHT POSITION"
```

Different computers set aside memory space for string variables in varying ways. Atari BASIC and Apple Integer BASIC both require that you tell the computer in advance exactly how many bytes to set aside for each string variable you plan to use. This is done with a DIMension statement such as

```
10 DIM A$(20), X$(110), RR$(3)
```

which then allows you to assign strings of up to 20, 110, and 3 characters to these three variables. Notice that in this case the DIM statement is not being used to specify the dimensions of an array, but rather to set an upper limit on the number of characters which may be assigned to a particular string variable at any one time. While the Atari has no limit to the size of a string DIMension, the Apple does limit you to 255 characters.

Neither Applesoft nor S-80 BASIC require this kind of string-length specification. In these BASICs, string variables may be assigned a string of any length from 0 through 255 characters. The S-80 does, however, require that you set aside a certain block of memory for string space — not to specify exactly how long each string is going to be, but to reserve enough total memory for all the string variables you plan to use. This is done with a statement such as

```
CLEAR 1000
```

which specifies 1000 bytes to be reserved. If no CLEAR statement is used, 50 bytes are automatically reserved by the computer. At the same time, the CLEAR statement assigns zeroes to all existing numerical variables and the null string (no characters) to all existing string variables, and "un-dimensions" all arrays.

Incidentally, CLEAR may also be used without a number following it, both in S-80 BASIC and in Applesoft. In this case its effect is to do the clearing of variable values and dimensions, without (in the case of the S-80) affecting the amount of string space reserved. The same variable-clearing function is performed by the statement CLR in Atari BASIC or Apple Integer BASIC. (However, CLR can be used only as an immediate command on the Apple, not as part of a program.) Because of their drastic effect on variables, the CLEAR and CLR statements are normally found at the very beginning of a program, before any variables at all are used.

Once you have a string, then, what can you do with it? Or rather, what can you make the computer do with it?

continued on next page

continued from previous page

Among the three computer systems we're discussing, there are five different dialects of the BASIC language: S-80 Level II and Disk BASICs, Atari BASIC, and Apple Integer and Applesoft BASICs. These divide themselves into two distinct groups according to the ways in which they handle strings: Atari BASIC and Apple Integer BASIC in one group, and the S-80 BASICs and Applesoft in the other group. Let's tackle Atari and Apple Integer first.

As we've seen, these BASICs require string variables to be dimensioned according to the maximum length of strings to be assigned to them. Consider the following program lines:

```
100 DIM X$(20)
110 X$ = "ABCDEFGG"
120 PRINT X$
```

Line 100 reserves 20 bytes for the contents of the variable X\$, line 110 assigns it a seven-character-long string, and line 120 PRINTs that string on the screen. But suppose that you wanted to PRINT only the first letter of X\$, or maybe the last one, or even the second through fourth ones. It's very easy to do, using the following statements in place of the one in line 120:

```
PRINT X$(1,1)
PRINT X$(7,7)
PRINT X$(2,4)
```

The two numbers in parentheses specify the first and the last characters to be PRINTed, counting from the left of the string. These three statements will PRINT "A", "G", and "BCD" when X\$ is defined as above. If you want to PRINT all the characters from a certain one through the end of the string, the second number can be omitted: X\$(3,7) and X\$(3) are equivalent, both returning the characters "CDEFG".

If you don't want to PRINT the characters in question, but rather assign them to another string variable, it's done in exactly the same way:

```
120 DIM Z$(20), LL$(50)
130 Z$ = X$(1,3)
140 LL$ = X$(5)
```

This will assign the characters "ABC" to the variable Z\$, and "FG" to the variable LL\$.

Strings can be not only broken apart, but also tied together. This is called "concatenation," and is accomplished in Atari BASIC and Apple Integer BASIC using statements

such as the ones in lines 220 and 230:

```
200 X$ = "ABCDEFGG"
210 Z$ = "HIJKL"
220 X$(8) = Z$
230 X$(13) = "MNOP"
```

Line 220 concatenates Z\$ onto the end of X\$, by specifying that the characters in Z\$ are to be added onto X\$ starting at the eighth position. This makes X\$ twelve characters long. Line 230 then adds four more characters, starting at the thirteenth position, for a total length of 16.

What, you may ask, would happen if line 230 were changed to

```
230a X$(10) = "MNOP"
```

(Note that the lower-case "a" following the line number is NOT a valid part of the line number; it's just used here to distinguish it from the original line 230.) What would happen to the characters "J", "K", and "L", which were the 10th, 11th, and 12th characters of X\$? The fact is that they would be replaced by the new characters "M", "N", and "O", followed by the remaining character in the string, "P". X\$ would then end up being 13 characters long: "ABCDEFGHIMNOP".

And what, on the other hand, would happen if you tried this:

```
230b X$(16) = "MNOP"
```

In this case you're trying to add four characters onto a 12-character string, starting at the 16th position. Does that make sense? Well, it doesn't make sense to the Apple (you'll get a ***STRING ERR message), but it does to the Atari. Atari BASIC will allow you to add a sub-string anywhere within the total dimensioned length of the string; any gaps between assigned characters (such as the 13th through 15th positions in this example) will be filled in with other characters. PRINTing X\$ after executing this new line 230b would display "ABCDEF GHIJKL@@@MNOP" where the "@@@" might be just about any characters but typically would be little graphic hearts.

In the same way, the Atari allows you to replace any characters in the middle of a string with other characters, using a statement like

```
240 X$(3,5) = "ZZZ"
```

This will replace the "CDE" in X\$ with "ZZZ", without affecting the rest of the string. (If the statement were X\$(3) = "ZZZ", the last "Z" would

become the final character of the new string, which would then be just five characters long.)

It's in these two areas of string splitting and string concatenation that the two groups of BASICs we're considering work very differently. Before discussing other string functions which all the systems have in common, then, let's consider how Applesoft and the S-80 BASICs handle these types of manipulations.

In this case the concatenation of strings is very similar to the addition of numbers. The equivalent of the original lines 200-230 above would look like this:

```
200 X$ = "ABCDEFGG"
210 Z$ = "HIJKL"
220 X$ = X$ + Z$
230 X$ = X$ + "MNOP"
```

Line 220 concatenates Z\$ onto the end of X\$, and then assigns the new, longer string to X\$; line 230 does the same with the characters "MNOP". This approach does not require that you know how long X\$ is before adding sub-strings to it, as the other approach does. On the other hand, if you wanted to change some of the existing characters in X\$ (as in lines 230a and 240), the procedure would be more complicated than with the Atari, involving the use of one or more additional string functions.

Speaking of additional functions, take a look at the following program:

```
300 ZZ$ = "ABCDEFGHII"
310 L$ = LEFT$(ZZ$, 1)
320 R$ = RIGHT$(ZZ$, 4)
330 M$ = MID$(ZZ$, 5, 3)
340 PRINT L$, R$, M$
```

LEFT\$, RIGHT\$, and MID\$ are not string variable names, but rather string functions or operators. They are capable of isolating one or more characters from the left end, right end, or middle of a string with the greatest of ease. The PRINT statement in line 340 will display the following on the screen:

```
A FGHI EFG
```

In words, here's what lines 310-330 do. The statement L\$ = LEFT\$(ZZ\$,1) picks the leftmost 1 character out of ZZ\$ and assigns that character to L\$. The statement R\$ = RIGHT\$(ZZ\$,4) picks the rightmost 4 characters out of ZZ\$ and assigns them to R\$. And the statement M\$ = MID\$(ZZ\$,5,3) picks 3 characters out of the middle of ZZ\$, beginning with the 5th character of the string, and assigns them to M\$.

Notice that the LEFT\$, and RIGHT\$ functions have two "arguments" while MID\$ has three. The arguments must be in the order indicated, enclosed in parentheses and separated by commas. Arguments can be constants, variables, or even expressions which use other constants, variables, operators, and functions. The following are all valid expressions (assuming that the computed values of the arguments are within the legal range for the given string):

```
360 Q$ = MID$(A$, N-1, 3)
365 S$ = MID$("ABCDEFGHIJ",
G*M1-M2, AA)
370 R$ = LEFT$(C$+D$, X/10)
```

In addition to chopping and joining strings, there are a few other very useful things that computers can do with them. All the BASICs under consideration have a function which gives the LENGTH of any given string; it's used in this way:

```
400 L = LEN (A$)
```

While the previous string functions return a string value, the LEN function returns a number value, which is greater than or equal to zero (and less than or equal to 255, except in the Atari's case). One very common use of this function is to set the limit on a loop, which might be created, for example, to examine each character of a string one by one. If you know ahead of time how long a string is going to be, that's fine; but often you don't, and then you really need the LEN function. The following example searches for the letter "A" in any given string and PRINTs a message each time it finds one:

```
500 FOR J = 1 TO LEN(X$)
510 C$ = MID$(X$, J, 1)
520 IF C$ = "A" THEN PRINT "I
FOUND ONE!"
530 NEXT J
```

A nice feature of S-80 Disk BASIC is its INSTR function, which works in somewhat the same way as the above loop does. It's used to search for any substring of one or more characters within another string. Unlike the above loop, however, INSTR will find only the first occurrence (if any) of the substring. An example of its use would be

```
540 P = INSTR (1, X$, "A")
```

where P will be assigned a number equal to the starting position of the

substring ("A") within the larger string (X\$). The initial number within the parentheses specifies the position in the larger string where the search is to begin; if it's omitted, it defaults to 1. If the substring is not found, INSTR returns a value of zero.

There are a couple of pairs of complementary functions which are often very helpful in working with strings. The first of these pairs consists of the ASC function and the CHR\$ function. ASC returns the ASCII value of a character, while CHR\$ returns the character corresponding to a specified ASCII value. (But CHR\$ is not available in Apple Integer BASIC.) The command

```
PRINT ASC ("A")
```

will display the number 65 on the screen, for example. Within the parentheses there must be either a character enclosed in quotation marks, or a string variable name. If a string is specified which has more than one character, the number returned by the function will be the ASCII value of the leftmost character of the string.

The CHR\$ function does the reverse, and allows you to manipulate characters which may not even be accessible from the computer's keyboard. On the S-80, for example, the block-graphics characters cannot be typed in from the keyboard, but can all be PRINTed to the screen using statements such as PRINT CHR\$(150);CHR\$(188). For another example, Applesoft allows you access to the underline character, the backslash, and the right bracket through the CHR\$ function: Try PRINTing CHR\$(91), CHR\$(92), and CHR\$(95).

This is also the way to include in strings, or to PRINT on the screen, normal quotation marks WITHIN a string. An assignment statement such as

```
B$ = "SAY "HELLO"!"
```

will give an error message. But B\$ can be assigned the desired value using this strategy instead (where 34 is the ASCII value of a quotation mark):

```
550 B$ = "SAY " + CHR$(34) +
"HELLO" + CHR$(34) + "!"
```

If you plan to use the double-quote character (or any other inaccessible character) a lot, it helps to assign it to a

variable and then use the variable in place of the CHR\$ function. These lines could replace line 550, above:

```
550 Q$ = CHR$(34)
555 B$ = "SAY " + Q$ +
"HELLO" + Q$ + "!"
```

The second pair of complementary functions consists of the VAL function and the STR\$ function. (Neither is available in Apple Integer BASIC.) The operation of VAL was discussed last month; its function is to try to evaluate the characters of a string as a numerical value. The lines

```
600 A$ = "9804"
610 B$ = "-043NPL"
620 X = VAL (A$)
630 Y = VAL (B$)
```

assign the numbers 9804 and -43 to X and Y. The STR\$ function does just the opposite: It makes a number into a string. Here's an example:

```
650 NUM = 62473.88
660 N$ = STR$(NUM)
```

This isn't used too much, but can be very helpful in formatting screen displays, building mixed number-and-letter strings which serve to store various kinds of data, and other miscellaneous purposes.

Finally, the S-80's handy STRING\$ function should be mentioned. It allows you to PRINT a whole lineup of any character you specify, with a single command. These two lines would have identical results:

```
700 PRINT
"++++++"
"++++++"
700 PRINT STRING$(25,33)
```

The first number in the parentheses is the number of characters to PRINT; the second is the ASCII value of the character, or the character itself. If your chart of ASCII values isn't nearby, or if you want to make the STRING\$ statement self-explanatory when you look at it later, you can do something like this to PRINT a full screen line of periods:

```
720 PRINT STRING$(64, ".")
```

So much for this month's Encounter. If you're not still totally strung out then, I hope you'll be back. I will, anyway.

TRS-80 Model I and Model III Business and Communications Software

Business accounting systems. Complete Package \$750.00
A/R, A/P, P/R & G/L

Accounts receivable: Price \$195.00

This self configuring system can be used on any 48K System with one to four disk drives. The system can be run coordinated with General Ledger or it can be used by itself. The user can decide how many disks he needs to accommodate his system & configure it to his requirements. Up 999 Customers & 999 invoices can be handled on a three disk system. A utility menu function allow for verifying programs & files, initializing files & configuring the system. As the users requirements change, the system can be reconfigured to accomodate the large capacity. The system supports NEBS Billing Statements, aging analysis reports, customer activity reports, unbilled report, open & close invoice reports. Sales tax is automatically handled & up to 9 other tax or sales categories are also support. This sytem is time tested in over 4,000 user accounts.

Accounts Payables: Price \$195.00

This system, like all the Mod-III systems, is user configurable. Up to 999 Vendors & 999 invoices can be handled on a three disk system. The user can also coordinate this system with the General Ledger system & eliminate many of the data entry chores of the Ledger system. NEBS computer generated payable checks, aged analysis, Vendor activity & open & closed invoice reports and other reports are included in this package. This system has been run on Wang computer for several years with wide user approval.

Payroll: Price \$195.00

The payroll system is also user configurable & can run on from one to four disks. Up to 250 employees living in up to 9 states can be paid using this system. NEBS computer payroll checks are supported. This system supports a wide range of reports including, W-2's, absentee, deduction, check register, payroll history, and more. Weekly or biweekly pay periods; salary and hourly pay is supported. Again, this system can also feed Ledger & reduce the data entry requirements for your General Ledger.

General Ledger: Price \$195.00

This system is designed to interface with the other accounting systems or it can be run by itself. The user can define his own Chart of Accounts & report formatting. Trial balance, & income statements as well as user definable special reports are all included as part of the system.

Complete package for Model-I (#25-229020D)	\$750.00
Complete package for Model-III (#27-229020D)	\$750.00
Accounts Receivable for Model-I (#25-229010D)	\$195.00
Accounts Receivable for Model-III (#27-229010D)	\$195.00
Accounts Payable for Model-I (#25-229011D)	\$195.00
Accounts Payable for Model-III (#27-229011D)	\$195.00
Payroll for Model-I (#25-229011D)	\$195.00
Payroll for Model-III (#27-229011D)	\$195.00
General Ledger for Model-I (#25-229011D)	\$195.00
General Ledger for Model-III (#27-229011D)	\$195.00
ST80-III for Model-I (#25-229001D)	\$150.00
ST80-III for Model-III (#27-229001D)	\$150.00
HOST package for Model-I (#25-229002D)	\$50.00
HOST package for Model-III (#27-229002D)	\$50.00

ST80-III The ULTIMATE Communications Package: Price \$150.00

This is our top of the line communication package. Full disk support including DOS commands have been implemented. ST80-III has been on the market for over two years and has become the standard in TRS-80 communication. This package has been used in a wide variety of applications including use with: Addresso-multigraph, Compugraphics, Spectrometers, and a wide range of Time-sharing computers.

IBM	CDC	ITEL	Honeywell
DEC	WANG	Prime	Data General
Amdhal	RCA	XEROX	GE
Apple	Heath	Northstar	Altos
Superbrain	PET	Cromemco	HP 2000

The package includes the ST80-III smart terminal program & nine other communication utilities: Fully documented with easy to follow instruction, ST80-III is by far the best terminal product on the market today. Features:

- 1) User configurable communication tables
- 2) Auto Logon
- 3) Last line repeat
- 4) Formatted video (Page, Scroll & Formatted)
- 5) Direct cursor addressing
- 6) File transfer from disk or to disk
- 7) Printer support
- 8) Echo, Feedback & Veriprompt (tm) verifies data transmitted
- 9) 110 to 9600 BAUD support
- 10) Remote control of,Memory open & close, Printer on & off Video on & off & auto logon
- 11) Help display
- 12) User definable function keys

HOST communications: Price \$50.00

This program is by far the best HOST program you can buy. It supports the PRINT @ statement for the remote TRS-80 running any of the ST80 smart terminal programs. All of the ST80-III advanced functions are supported by host allowing easy access via Basic, Fortran and machine language programs. Host features include:

- 1) User defined RS-232 port addressing
- 2) Definable BAUD rates from 110 to 9600
- 3) Definable break (yes/no)
- 4) Allow line feeds
- 5) Commands:
 - a) Turn on TRS (request to send),
 - b) Turn off TRS,
 - c) Receive data only from terminal,
 - d) Receive data only from host,
 - e) Send data only to host,
 - f) Send data only to terminal,
 - g) Operate in dumb terminal mode,
 - h) Operate in ST80 mode,
 - i) Check CTS status.

This is a self relocating subroutine that can load anywhere in high memory.



TSE-HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

LEYTE

by Victor A. Vernon, Jr.

Apple and Atari translations by Alan J. Zett

"Leyte" is a sea-battle simulation for a 16K S-80, Atari, or Apple (with Applesoft.)

This simulation is based on the "Battle of Leyte Gulf" which was fought in October, 1944. The following introduction provides a background and a brief synopsis of the events surrounding this great naval battle. continued on next page



continued from previous page

After the U.S. Central Pacific forces had taken the islands of Pelelu and Angour, and MacArthur's Southwest Pacific forces had secured New Guinea, President Roosevelt met with Admiral Nimitz, who commanded the Central Pacific (CINCPAC), and General MacArthur. At that meeting, MacArthur convinced Roosevelt that the next objective should be the liberation of the Philippines. Admiral Nimitz wanted to bypass the Philippines and invade the island of Formosa. Loss of either the Philippines or Formosa would tear the Japanese empire in two, cutting off the home islands from the oil and wealth of the Indies.

The American plan was to invade the Island of Leyte in the center of the Philippines. The invasion force would be the Eighth Army under the command of MacArthur. The Seventh Fleet, commanded by Admiral Kinkaid, would transport the troops to the beach. Admiral Halsey's Third Fleet would provide protection from any attempts by the Imperial Fleet to disrupt the landings.

This would be the first time that both American Pacific Fleets would operate together. Heretofore the Third Fleet was in the South Pacific. (The "other" American Fleet, the Fifth, was in fact the Third. Only the commander was different: Admiral Spruance commanded the "Fifth Fleet.") The reason for the joint venture was the composition of the Seventh Fleet and the distance it would have to travel to perform its task. Before Leyte, MacArthur always had the luxury of establishing a beachhead within range of his land-based Fifth Air Force. At Leyte he would not have this support until his engineers could build bases. Until then the Seventh Fleet would not be able to defend itself against the Japanese naval and air power.

The Seventh Fleet was mainly an invasion fleet. It contained troop transports and LST's. The only capital ships were in a bombardment group consisting of old battleships (some veterans of Pearl Harbor,) a few cruisers (some Australian), and destroyers with escorts. There were 18 escort carriers, which may at first seem like a large number of carriers. But

when one considers that they averaged only 28 aircraft each, and those aircraft were F4F's, it can be seen that they were not prepared for any large-scale naval engagements — especially since they carried no armor-piercing bombs and no torpedoes.

The Third Fleet, however, was a true battle fleet. It contained the new Essex class carriers, each with 100 aircraft. It also had several light carriers with 60 to 80 aircraft each. Support was provided by the new Iowa class battleships as well as heavy cruisers, light cruisers, and destroyers. The Armada created by the combined Third and Seventh fleets was, until then, the largest invasion fleet ever assembled. Only the fleet which attacked Okinawa in April, 1945, was larger. (While the Normandy landing fleet was made up of ships from several nations, its size did not match either Leyte or Okinawa.)

Against them the Imperial Japanese Fleet was woefully outclassed. Only superior strategy could carry them to victory and force the U.S. to accept a negotiated peace. The Japanese knew from the start of the war that they could not win the war; all they wanted was to win an agreement which would preserve as much of their empire as possible. After the defeat and almost total destruction of their carrier forces in the First Battle of the Philippine Sea, the Japanese developed a plan to defend the inner portion of the empire. This plan was called the SHO-1 plan. Its strategy was to draw the Americans "out on a limb," and then to cut off that limb. With the Leyte landings they were now prepared to perform the surgery.

The SHO plan was based on two very important facts:

1. The Imperial Fleet was no match for the Third Fleet.
2. The Seventh Fleet was no match for the Imperial Fleet.

In order to be victorious, the Third Fleet would have to be neutralized, thus leaving the Seventh at the mercy of the Imperial Fleet. To do this would require nearly perfect timing and a lot of luck. Amazingly, SHO-1 succeeded

in almost every detail. Only an error in judgment, perhaps caused by the vastness of the plan, denied the Japanese full victory.

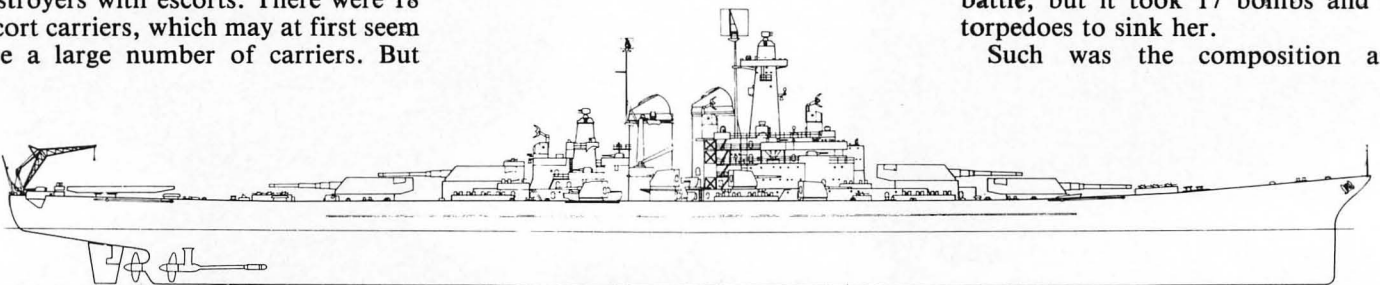
The Japanese forces were divided into three separate fleets: the Northern force, the Southern force, and the Central force. The Northern force comprised what remained of the carrier fleet, and was commanded by Admiral Ozawa. Short of planes, with untrained or partially trained pilots, its mission was to be sunk. If everything worked, this "bait" would lure Halsey and the Third Fleet north, away from Leyte, while the Central and Southern forces drove the American forces out of the Philippines.

The Southern force was actually made up of two groups; one was commanded by Admiral Shima and the other by Admiral Nashimura. While they were supposed to coordinate their movements, Shima and Nashimura were not on speaking terms; therefore they attacked separately. Their line of attack was to sail through the Surigao Strait, which led to the southern part of Leyte Gulf.

The Central force, commanded by Admiral Kurita, would be the main attack force. Comprising 5 modern battleships, 10 heavy cruisers, 2 light cruisers, and 15 destroyers, it would sail through the Sibuyan Sea, through San Bernardino strait, and then turn south around the island of Samar to enter Leyte Gulf from the north. They were to meet Shima and Nashimura early on October 25, 1944, and together wreak havoc upon the American transports in the harbor.

If Ozawa successfully lured Halsey to the north, then Kurita was to be the surgeon who would cut off the fragile limb supporting the Americans. Kurita's force was very powerful. Although he had no air support, he did have the two largest warships afloat, the Musashi and the Yamato. Each displaced 64,000 tons, and each carried nine 18-inch rifles. (Naval cannons are called either guns or rifles, but "rifles" is more correct.) It should be noted that the more famous German battleship Bismark had only 15-inch rifles; in fact, the Bismark could fit inside the Yamato. The Musashi was sunk in this battle, but it took 17 bombs and 20 torpedoes to sink her.

Such was the composition and



disposition of each of the fleets involved. Now let's review briefly the events of October 20 through 25, 1944. Remember that this was the largest naval battle ever fought; it covered thousands of square miles of ocean and involved every type of warship from the smallest P.T. boat to the largest battleship. If things become confusing and you would like to do additional reading, two very good books on this subject are Edwin P. Hoyt's "The Battle of Leyte Gulf" and Adm. Morrison's "Naval Operations in World War II."

On October 20, the American forces landed on the western shore of Leyte Island. Almost immediately, SHO-1 was put into operation. The landings were accomplished with only some land based air attacks, from Luzon, by the Japanese. Two days later, the American submarines Darter and Dace spotted Kurita's Central force in the Sulu Sea. They attacked, sank two cruisers, and sent news of Kurita's course and speed to Halsey and Kinkaid. In an attempt to maneuver for a second attack, Darter ran aground; during most of that morning the two submarines attempted to free her, but nothing worked and she had to be scuttled. The crew was taken off by Dace, which returned with both crews to Pearl.

At first light Halsey sent out his search planes; their orders were to find the Japanese. They found Kurita in the Subuyan Sea, sailing east toward San Bernardino Strait. Halsey immediately launched a full air strike.

About this time Halsey issued a very important directive, one which nearly determined the outcome of the battle. Halsey's directive was intended only for the Third Fleet but was picked up by Admiral Kinkaid. It concerned the formation of a new task group, commanded by Admiral Lee and comprising 6 battleships, 2 cruisers, and some screening destroyers. This task group was to guard San Bernardino Strait when (and if) Halsey found the Japanese carriers.

At this time Ozawa was trying his best to be found, but was not having any luck. He had broken radio silence as soon as he left home waters, but at-

mospheric conditions prevented Halsey's radio men from receiving the transmissions.

Halsey's strike found Kurita still in the Sibuyan Sea. The strike concentrated on the Musashi, sinking her; several other ships were also damaged. Kurita, lacking any air cover, decided to turn around. This action was interpreted by the American pilots as a full retreat. They reported to Halsey that Kurita's fleet had been destroyed and was in full retreat. By this time Ozawa had been discovered, and Halsey made a fateful decision. Faced with the decision to stay and guard the strait or to head north and attack Ozawa, he decided that his mission was offensive and not defensive. He headed north and, because he thought Kurita was no longer a threat, took Lee's battleship force with him. He informed Kinkaid that he was heading north. But because he did not know that Kinkaid knew about Lee's group, he did not tell of his intentions of taking Lee along. Kinkaid assumed that Lee was still guarding the strait and, since he was about to engage the Japanese Southern force, did not deploy any units to the northern part of Leyte Gulf.

The night of October 24-25 was a very eventful one. Halsey was heading north towards Ozawa, while Ozawa was heading south towards Halsey. Kurita had turned east again, and was heading for the mouth of San Bernardino Strait. Shima and Nashimura were sailing north through Surigao into a trap. Kinkaid's bombardment group, commanded by Admiral Olendorf, was moving into position to stuff the bottleneck of Surigao Strait. P.T. boats patrolled the southern end of Surigao; along the sides were destroyers; at the northern end was a floating dam of cruisers; and behind the cruisers were battleships. The Japanese, having lost the element of surprise, and with no hope of coordinating their attack with Kurita, sailed on through the night into the trap. Because of the narrow waters, the Japanese were forced to sail in a single-file line. One by one they sailed

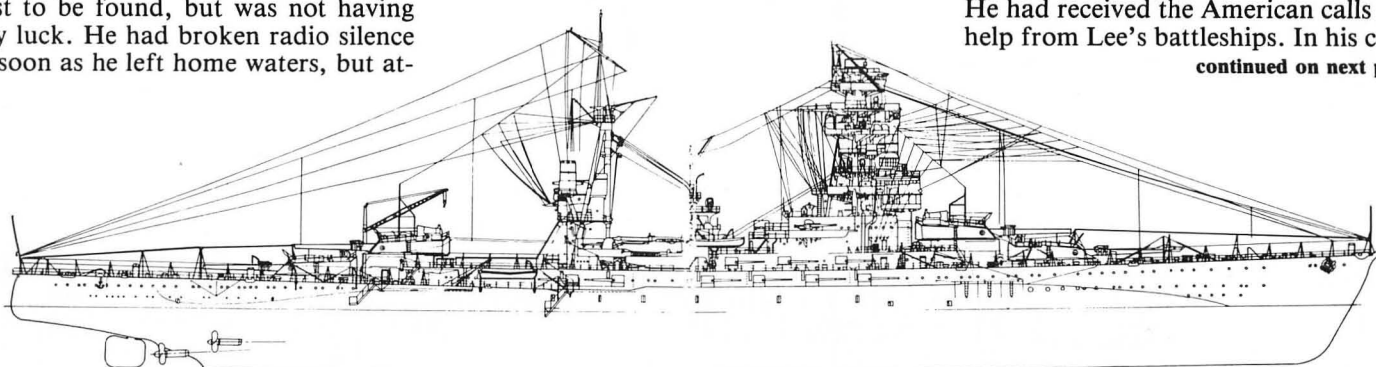
up the gauntlet. One by one they were destroyed.

October 25, 1944, was to become one of the most crucial days in the Pacific War. At daylight, Halsey was within striking distance of the Northern force and Olendorf was busy mopping up what remained of the Southern force. In the middle was the San Bernardino Strait, an open, unguarded waterway. Through this strait, at 20 knots, steamed Kurita and the Central force. Should these 22 ships reach Leyte they would turn the landings into bloody shambles, with Halsey too far north and Olendorf too far south. This approach to Leyte was guarded only by light American units consisting of small escort carriers, destroyers, and destroyer escorts.

The Japanese were first sighted by surprised, horrified pilots from the escort carriers. Admiral Sprague, who commanded the escort carriers, immediately ordered his thin line of destroyers to attack. These sailors sacrificed themselves for Leyte. He also started calling for help "in the clear;" that is, uncoded. (If Kurita didn't know where he was then, Kurita was blind!) Barrage after barrage straddled the American carriers, protected only by the intrepid destroyers. The destroyers laid a screen of real and chemical smoke between the Japanese and the escort carriers, and then bore in to deliver a furious torpedo attack — one of the most frantic, desperate actions in the war at sea. Every plane that could get off the carriers joined the unequal fight. Trained only to support ground troops, the jeep pilots hit the Japanese with what they could, with all they had. But .30-caliber machine guns do very little damage to 16-inch armor-plate.

The Japanese kept pounding away, smashing the reeling escorts and the dauntless destroyers. But the reckless American attack continued, scattering and confusing the Japanese. Kurita, especially, was in a quandary. He knew that the Southern force had been destroyed; he knew nothing of Ozawa's success in luring Halsey away. He had received the American calls for help from Lee's battleships. In his con-

continued on next page



continued from previous page

fused state he thought that the escort carriers were the fleet carriers of the Third Fleet, that the destroyers were cruisers, and that the destroyer escorts were large Fletcher class destroyers. The only possible conclusion he could draw was that he was sailing into a trap with all that remained of the Imperial Fleet.

At this point Kurita held history in the palm of his hand. SHO-1 had worked! Kurita could have smashed the Leyte landings and totally isolated those American troops already on the beach. He could have forced the Americans to accept a negotiated peace, favorable to the Japanese. He could have saved the Empire. But what did he do? He quit. He turned around again, this time for good.

Five American ships were sunk: Gambier Bay, St. Lo, Johnston, Roberts, and Hoel. There were 2800 casualties. In the next days Halsey's airmen further smashed Kurita's ships as they retreated to Japan.

The Japanese plan worked, but it availed them nothing. All three of their forces were beaten. In four days of far-flung fighting they lost 1 large carrier, 3 light carriers, 3 battleships, 6 heavy cruisers, 4 light cruisers, and 9 destroyers — 26 of their finest ships. The once-magnificent Imperial Fleet was no more. That dream was over. It died in the Battle for Leyte Gulf, and it died in vain.

The preceding narrative explains what actually happened. The following will explain how to play the simulation. It is based on what MIGHT have happened if Lee had been waiting for Kurita at the mouth of San Bernardino. You will be Lee, the computer will be Kurita. The American fleet will consist of the battleships Jersey, Washington, Massachusetts, and Alabama, the cruisers Baltimore and Pittsburgh, and 6 Fletcher class destroyers. The Japanese fleet will be the battleships Yamato, Nagato, Kongo, and Haguro, the heavy cruisers Chokai and Sizuya, the light cruiser Nagara, and 8 destroyers.

In both cases the destroyers are considered to be a collective force. That is, the target for one is the target for all. While individual destroyers can be targeted and sunk, all of them must be sunk to destroy their force.

The screen displays a map showing the Island of Samar, with San Bernardino to the left and Leyte Gulf to the right. North is the left side of the screen, east is the top, south is the right and west is the bottom. The Japanese fleet will move from left to right, from San Bernardino to Leyte. To win you must sink all 7 capital ships: everything larger than a destroyer. If any get to Leyte or if all your ships (including destroyers) are destroyed, then you lose. Of course, the Japanese will NOT turn around.

You can issue any of these four commands, by pressing the appropriate number key:

- 1 - status report
- 2 - fire guns
- 3 - fire torpedoes
- 4 - course correction

The status report will tell you the condition of each of your ships; you will not be told the condition of the Japanese ships. (The computer will tell you only if they are sunk, at the time when it checks the status of all ships.)

If you choose to fire your guns you will then be asked for a target for each of your ships. The Japanese ships are numbered as follows; use the number to designate a target:

- 1 - Yamato
- 2 - Nagato
- 3 - Kongo
- 4 - Haguro
- 5 - Chokai
- 6 - Suzuya
- 7 - Nagara
- 8 - Destroyers (collectively)

If a target is sunk you will be directed to select another. Whether or not you score a hit is determined by three factors:

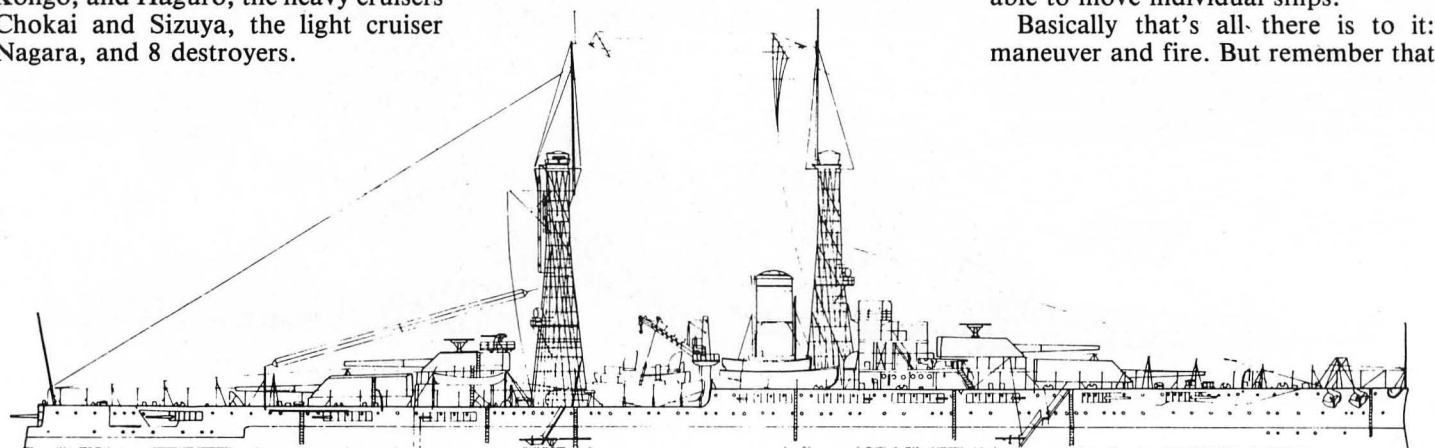
- 1 - a randomly selected number
- 2 - how close you are to the target
- 3 - whether the target has been fired upon by another ship.

The closer you are to the Japanese, the better will be your chances to hit something. The third factor comes into play because naval gunfire is accomplished by first firing a round, and then correcting the aim by noting where the first shells land. This can be done by watching the splashes; but if two or more ships are firing on the same target, it is impossible to tell which splashes go with each ship's fire. The Japanese avoided this by coloring their shells, thus giving different colored splashes for each ship.

Firing torpedoes is essentially the same, except that only destroyers carry torpedoes; thus, you will only be able to fire at one target. Also remember that while a torpedo can do much more damage than a shell, the Mark 14 torpedoes used by the U.S. Navy in World War II were notoriously unreliable. At times they would sink; at other times they would breach and explode against a wave. If they did hit a ship, they didn't always explode. The Japanese "long Lance" torpedo was larger (24 inches versus 21), and it WORKED.

Command #4 is a course correction; it is not necessary to access this command on every turn. Your course is initially set in a northerly direction. To change this you will be prompted to enter a number which will lead you in any one of 8 directions. The Japanese fleet will be heading generally south at a flank speed, so that if they get past your fleet, you will not be able to intercept them. Also remember that you are a Fleet Commander, not a Ship's Captain. Your orders will be for the fleet as a whole, and you will not be able to move individual ships.

Basically that's all there is to it: maneuver and fire. But remember that



maneuvering is as important as firing. If you stay too far away from the Japanese, your chances of sinking anything will be very slim, and they will sail on to Leyte while you watch them go by. But if you get too close too soon, the Japanese will blow you out of the water with their superior gunfire and torpedoes (Japanese cruisers also carry torpedoes.)

Your position on the map will be indicated by an "A," and the Japanese position by a "J." If both fleets are at the same place you will see an "*".

VARIABLES:

A: Location of American fleet.

A(n,nn): Holds all pertinent information about the American fleet.

A\$(n): Names of American ships.

A1: Course of American fleet.

AZ: Miscellaneous.

C: Movement counter. Determines when computer will check on status of all ships. Also helps determine if Japanese fire guns or torpedoes.

G: Command variable.

H: Hit counter.

H(n,nn) or H(n,nn,x): Hit table; determines how much damage is done if a hit is scored. Damage is determined by the size of the shell, where it hits, and what it hit.

I: Used in POKEing and PEEKing screen memory.

I\$: Keyboard input variable.

J: Position of Japanese fleet.

J(n,nn): Holds all pertinent information about the Japanese fleet.

J\$(n): Names of Japanese ships.

JT: Number of reserve torpedoes for Japanese fleet.

M: Modifier; changes range of Z, depending upon fleet locations.

M1: Temporary storage for M.

NT: Number of reserve torpedoes for American fleet.

R\$: Used in status report messages.

T: Used in arrays to signify target.

T\$: Keyboard input variable.

T1,T2: Temporary storage variables.

T1\$,T2\$: Temporary storage variables

Because of the extensive line-by-line documentation provided by the author, and the fact that the documentation is essentially the same for all three versions, we have not spliced it into each separate listing. Rather, it is presented below in a way that makes it applicable to all three programs. Note that we've indicated RANGES of line numbers; you won't find every named line in every version.

In fact, lines such as 99 and 149 won't be found in any of the versions; they simply indicate the upper limit of that range of lines.

Line 50: READ/DATA cure for some S-80 Model I computers.

Lines 60-99: Initialization.

Lines 100-114: Loads the arrays.

Lines 115-149: Draws the map on the screen. Map cannot be redrawn during the program.

Lines 150-189: Moves the fleets.

Line 150: Sets half-turn & hit counters to 0.

Line 160: Clears bottom of screen, puts a "." at current positions.

Line 165: Reads new Japanese position, checks to see if Japanese have reached Leyte.

Lines 170-174: Sets random number modifier to maximum, then moves Japanese fleet 1 above or 1 below center. This varies the route the Japanese will take from game to game.

Lines 175-179: Prevents American fleet from hitting land or going off screen.

Line 180: Puts both fleets in position.

Lines 190-279: Command routine.

Lines 190-239: Prints commands & changes modifier.

Line 240: Puts time limit on command entry.

Line 250: Checks for an input of 1, 2, 3, or 4.

Line 260: Branches to selected command.

Line 270: Changes half-turn count if time limit is exceeded.

Lines 280-479: Displays the status of American ships.

Line 280: Starts loop & checks if ship is sunk.

Line 290: Branches if ship is Iowa or New Jersey.

Lines 300-339: Checks status of Iowa & New Jersey.

Line 340: Checks if ship is cruiser or destroyer.

Lines 350-379: Checks status of other battleships.

Line 380: If X = 7 then ship is destroyer.

Lines 390-419: Checks status of cruisers.

Line 420: Checks status of destroyers.

Lines 430-469: Displays the ships' status.

Line 470: Goes back to command display. This was not made a subroutine to enable player to check the status without limiting his time to decide on his next command.

Lines 480-619: U.S. gunfire routine.

Line 480: Changes half-turn count & skips any sunken ships.

Lines 490-519: Prompts for a target. Rejects any non-numerical input.

Line 520: Array is subscripted 0-7, so it subtracts 1 from target. Also changes modifier to make target harder to hit if ship was already fired upon in this turn.

Line 530: If target is sunk, selects another.

Line 540: Resets hit counter to 0, fires once

Line 530: If target is sunk, selects another.

Line 540: Resets hit counter to 0, fires once for each gun.

Line 550: V determines where target was hit. If the ship firing was a

battleship & the target is hit on the front or back turret or on the bridge, then the number of guns on the target is decreased by one.

Line 555: Timing delay.

Line 560: Changes defense factors on target.

Line 570: Displays the number of main armament hits.

Lines 580-619: Repeats process for secondary armament.

Lines 620-720: U.S. torpedo fire.

Line 620: Changes half-turn count & modifier.

Line 630: Makes sure U.S. has torpedoes.

Line 640: Selects a target.

Lines 650-689: Selects a target.

Line 690: Fires torpedo & checks for hit.

Line 700: Determines where ship was hit and changes strength tables.

Line 710: Displays number of hits.

Line 720: Goes to Japanese firing routines.

Lines 730-879: U.S. course change routine.

Line 880: Checks turn count. Japanese only fire torpedoes on second half-turn if range is close enough.

Lines 890-1015: Japanese gunfire routine.

Line 890: Checks for sunken ships.

Line 900: Makes sure there is something to shoot at.

Lines 910-919: Selects next target.

Line 920: Fires guns and checks for hit.

Lines 940-969: Main armament hits.

Lines 970-1014: Secondary hits.

Line 1015: Checks turn count.

Lines 1020-3019: Japanese torpedo firing routine.

Line 1020: Displays torpedo fire & changes modifier.

Line 1030: Checks if cruisers can fire torpedoes.

Lines 1040-2039: Fires destroyer torpedoes.

Lines 2040-3019: Fires cruiser torpedoes.

Lines 3020-4020: Status update routine.

Line 3030: Resets target flags. Sets sunken ship flags for Japanese & displays messages.

Line 3050: Checks if American ships are sunk.

Line 3060: Checks status of American destroyers. Resets targets for Japanese destroyers.

Lines 3070-3089: If an American destroyer is sunk, subtracts 1 from total afloat & decreases torpedo reserves by 10.

Lines 3090-4019: Repeats process for Japanese destroyers.

Line 4020: If Japanese fleet is not destroyed, goes to command routine.

Lines 4030-5999: End-of-game win/lose messages.

Lines 4030-4059: Win message.

Line 4060: Double-checks American ship status.

Lines 4070-5999: "You lose" messages.

Lines 6000-6999: Data section.

Lines 6000-6079: Data for all A & J arrays.

Lines 6080-6109: Data for hits table.

Lines 6110-6999: Japanese movement data.

Lines 7000-7100: On the S-80, this is an error trapping routine for catching FOR/NEXT loops that produce errors. In the Apple, these lines do not exist. On the Atari, this is an erase-to-bottom-of-screen subroutine.

continued on next page

The most important book ever published for the Apple.

The most comprehensive description of Apple II firmware and hardware ever published — all in one place.

What's Where in the Apple?

- Guides you — with a numerical Atlas and an alphabetical Gazetteer — to over **2,000** memory locations of **PEEKs**, **POKEs**, and **CALLs**.
- Gives names and locations of various **Monitor**, **DOS**, **Integer BASIC**, and **Applesoft** routines — and tells you what they're used for.
- Helps BASIC users to speed up their programs.
- Enables assembly language programmers to simplify coding and interfacing.

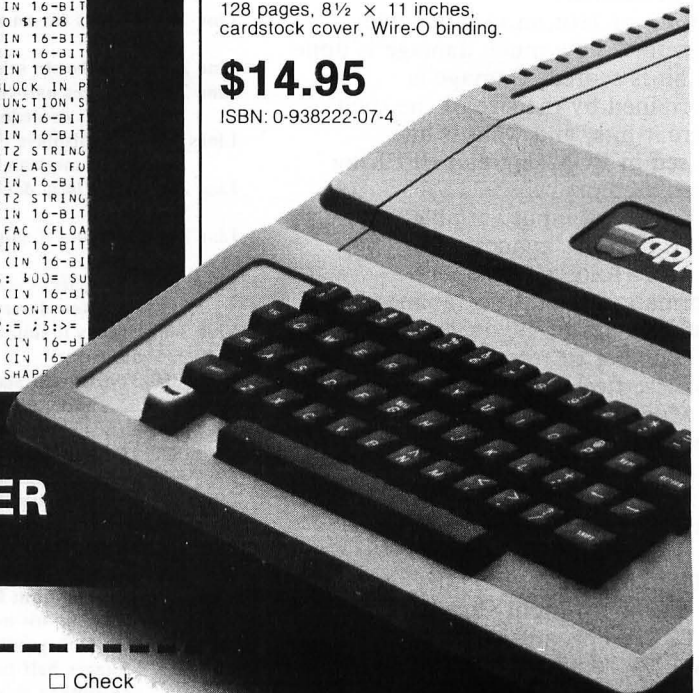
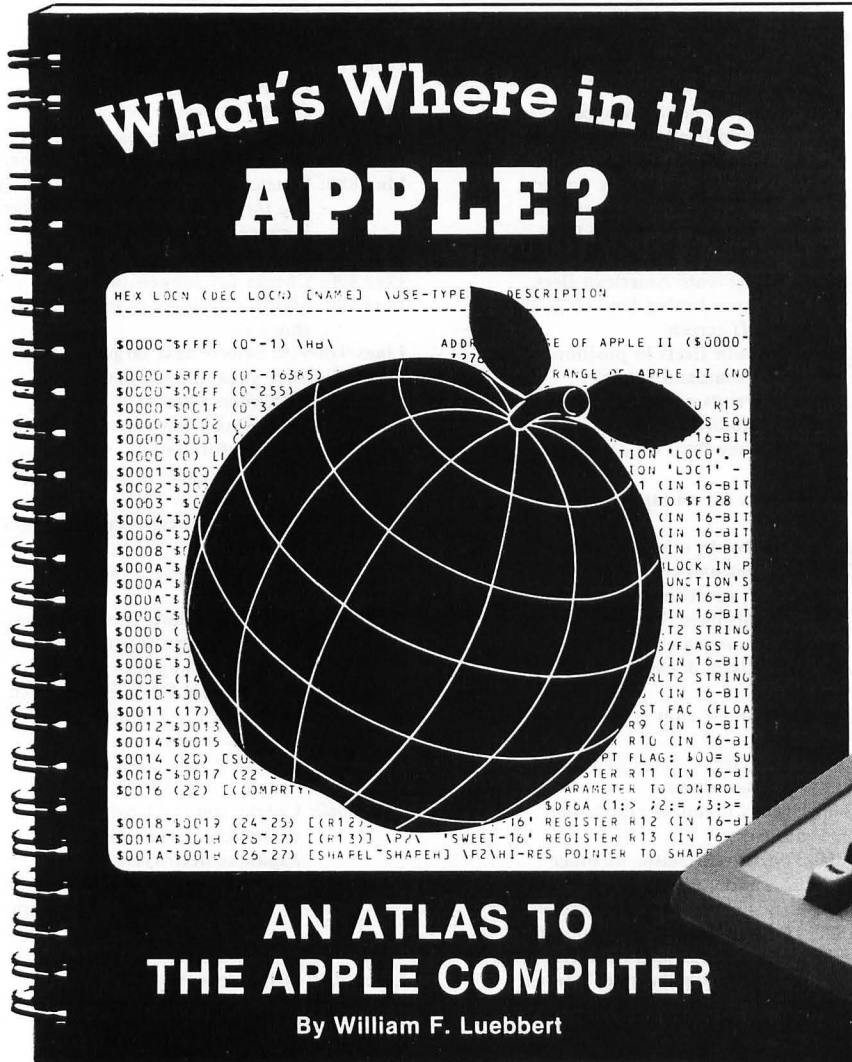
All Apple users will find this book helpful in understanding their machine, and essential for mastering it!

Ask for it at your computer store

128 pages, 8½ × 11 inches, cardstock cover, Wire-O binding.

\$14.95

ISBN: 0-938222-07-4



ORDER TOLL-FREE TODAY **800-227-1617** EXT. 564
(In California 800-772-3545 Ext. 564)

Yes! Please send me _____ copies of *What's Where in the Apple?* at \$14.95 each (in U.S. plus shipping).

Name _____

Address _____

City _____ State _____ Zip _____

Check for \$ _____ enclosed. (Add \$2.00 surface shipping for each copy.) Massachusetts residents add 5% sales tax.

VISA MasterCard

Acct. # _____ Expires _____

Signature _____

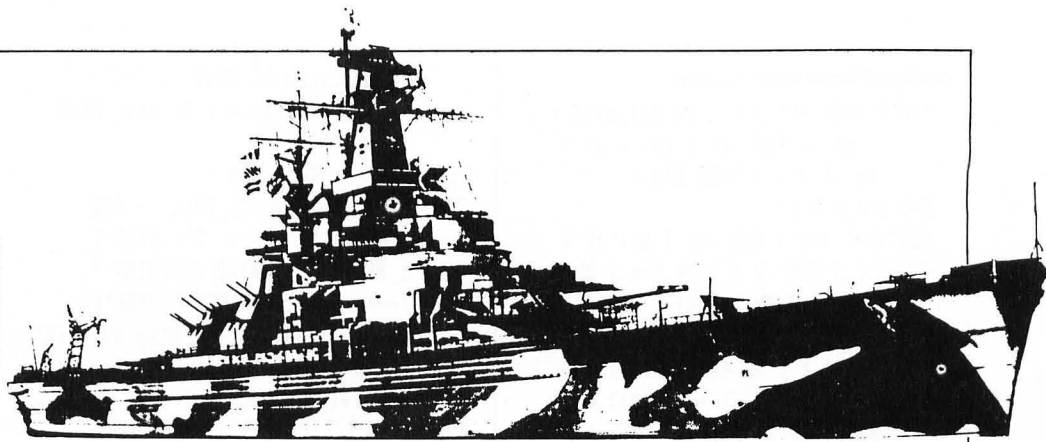
MICRO INK, Inc., 34 Chelmsford Street, P.O. Box 6502, Chelmsford, MA 01824

Apple Version

```

10 REM ORIGINAL VERSION BY:
20 REM VICTOR A. VERNON JR.
30 REM APPLE VERSION BY:
40 REM ALAN J. ZETT
50 DIM A(7,7),J(7,7),A$(7),J$(7)
   ,H(4,6,1):AZ = 40
60 DEF FN P(I) = 895 + ( INT (I
   / AZ) + 1) * 128 + ((I - INT
   (I / AZ) * AZ) + 1) - 984 *
   ((( INT (I / AZ) + 1) > 8) +
   (( INT (I / AZ) + 1) > 16))
70 TEXT : HOME : PRINT "ONE MOMEN
   NT PLEASE": PRINT "THERE IS
   A LOT OF DATA TO READ":NT =
   50:JT = 100:A = 167:A1 = -
   1:I = 0
100 FOR X = 0 TO 7: READ J$(X),A
   $(X): FOR Y = 0 TO 7: READ J
   (X,Y),A(X,Y): NEXT Y,X
110 FOR X = 0 TO 4: FOR Y = 1 TO
   6: FOR Z = 0 TO 1: READ H(X,
   Y,Z): NEXT Z,Y,X
115 HOME : FOR X = 1 TO 13: PRINT
   "***** RUN
   *****": NEXT X
120 INVERSE : HTAB 10: VTAB 8: PRINT
   SPC( 20): HTAB 8: VTAB 9: PRINT
   SPC( 24): HTAB 6: VTAB 10:
   PRINT SPC( 28): HTAB 5: VTAB
   11: PRINT SPC( 30): HTAB 2
   : VTAB 12: PRINT SPC( 35):
125 INVERSE : HTAB 1: VTAB 13: PRINT
   SPC( 39): HTAB 1: VTAB 14:
   PRINT SPC( 80): VTAB 14: PRINT
   " < SAN BERNARDINO STR.
   LEYTE GULF >": NORMAL
130 POKE FN P(371),211: POKE FN
   P(375),193: POKE FN P(379),
   205: POKE FN P(383),193: POKE
   FN P(387),210
150 FOR X = 1 TO 1: NEXT X:C = 0
   :H = 0
160 HTAB 1: VTAB 17: CALL - 958
   : POKE FN P(I + J),172: POKE
   FN P(I + A),172
165 READ J: IF J = 999 THEN 5010
170 M1 = 3:Z = INT ( RND (1) * 3
   ) + 1: IF Z = 1 THEN J = J +
   40
172 IF Z = 2 THEN J = J - 40
175 IF PEEK ( FN P(I + A + A1))
   = 32 THEN HTAB 1: VTAB 17:
   PRINT "YOU'RE ABOUT TO RUN
   AROUND!": PRINT "I WILL NOW
   CORRECT YOUR POSITION": FOR
   ZZ = 1 TO 1111: NEXT ZZ:A1 =
   - 40

```



```

176 IF A + A1 < 0 THEN A1 = 40
180 A = A + A1: POKE FN P(I + A)
   ,65: POKE FN P(I + J),74: IF
   A = J THEN POKE FN P(I + A
   ),106
190 HTAB 1: VTAB 17: PRINT "COMM
   AND ?": CALL - 958: IF J =
   A + 1 OR J = A - 1 OR J = A +
   40 OR J = A - 40 THEN M1 = 0
200 HTAB 1: VTAB 19: PRINT " 1 -
   STATUS REPORT": IF A = J THEN
   M1 = - 1
210 PRINT " 2 - FIRE GUNS": IF A
   = J + 39 OR A = J - 39 OR A
   = J + 41 OR A = J - 41 THEN
   M1 = 2
220 PRINT " 3 - FIRE TORPEDOES":
   IF A = J + 1 OR A = J - 1 OR
   A = J + 40 OR A = J - 40 THEN
   M1 = 0
230 PRINT " 4 - COURSE CORRECTIO
   N":M = M1
240 POKE - 16368,0: FOR X = 1 TO
   2000: IF PEEK ( - 16384) <
   128 THEN 270
250 GET I$: IF ASC (I$) < 49 OR
   ASC (I$) > 52 THEN 270
255 FOR X = 1 TO 1: NEXT X
260 G = VAL (I$): ON G GOTO 280,
   480,620,730
270 NEXT X:C = C + 1: IF C > 2 THEN
   150
275 GOTO 240
280 FOR X = 0 TO 7: IF A(X,5) <
   1 THEN R$ = "SUNK": GOTO 430
290 IF X > 1 THEN 340
300 IF A(X,4) < 50 OR A(X,2) < 7
   THEN R$ = "FLOATING JUNK YA
   RD": GOTO 430
310 IF A(X,4) < 100 OR A(X,2) <
   12 THEN R$ = "VERY HEAVY DAM
   AGE": GOTO 430
320 IF A(X,4) < 150 OR A(X,2) <
   17 THEN R$ = "MODERATE DAMAG
   E": GOTO 430
330 R$ = "ESSENTIALLY UNDAMAGED":
   GOTO 430
340 IF X > 4 THEN 380
350 IF A(X,4) < 50 OR A(X,2) < 7
   THEN R$ = "PREPARE TO ABAND
   ON SHIP!": GOTO 430
360 IF A(X,4) < 100 OR A(X,2) <
   14 THEN R$ = "SEVERE DAMAGE"
   : GOTO 430
370 R$ = "LITTLE OR NO DAMAGE": GOTO
   430
380 IF X = 7 THEN 420
390 IF A(X,4) < 20 OR A(X,2) < 4
   THEN R$ = "SINKING!": GOTO
   430
400 IF A(X,4) < 50 OR A(X,2) < 8
   THEN R$ = "HEAVY DAMAGE!": GOTO
   430
410 R$ = "LIGHT OR NO DAMAGE": GOTO
   430
420 IF A(X,4) < 25 OR A(X,2) < 5
   THEN R$ = "HEAVY DAMAGE!": GOTO
   430
425 R$ = "UNDAMAGED"
430 HTAB 1: VTAB 17: CALL - 958
   : PRINT "STATUS : "
440 PRINT A$(X): " : ";R$: IF X =
   7 AND A(7,5) > 1 THEN PRINT
   A(7,5)" AFLOAT"
460 FOR Y = 1 TO 1111: NEXT Y: NEXT
   X
470 GOTO 190
480 C = C + 1: FOR X = 0 TO 7: IF
   A(X,5) < 1 THEN NEXT X
490 HTAB 1: VTAB 17: CALL - 958
   : PRINT "ENTER TARGET FOR "A
   $(X):M = M1
500 GET T$:T = VAL (T$)
510 IF T < 1 OR T > 8 THEN VTAB
   18: PRINT "ENTER NUMBER OF T
   ARGET SHIP.": PRINT "CHECK I
   NSTRUCTIONS": GOTO 500
520 T = T - 1:M = M + J(T,7):J(T,
   7) = J(T,7) + .5
530 IF J(T,5) < 1 THEN PRINT J$(
   T)" SUNK": FOR ZZ = 1 TO 44
   4: NEXT ZZ: GOTO 490

```

continued on next page

continued from previous page

```
540 H = 0: FOR Y = 1 TO A(X,6):Z =
  INT (( RND (1) * 10) + 1) +
  M: IF Z > 4 THEN 570
545 H = H + 1
550 V = INT ( RND (1) * 5): IF X
  < 5 AND (V = 0 OR V = 1 OR
  V = 3) THEN J(T,6) = J(T,6) -
  1
555 FOR Z = 1 TO 444: NEXT Z
560 J(T,2) = J(T,2) - H(V,A(X,0),
  1):J(T,4) = J(T,4) - H(V,A(X
  ,0),0)
570 NEXT Y: IF H > 0 THEN HTAB
  1: VTAB 19: PRINT H" MAIN AR
  MAMENT HITS ON:": PRINT J$(T
  ): CALL - 958: FOR ZZ = 1 TO
  1111: NEXT ZZ:H = 0
580 M = M1: FOR Y = 1 TO A(X,6) *
  2:Z = INT (( RND (1) * 10) +
  1) + M: IF Z > 4 THEN 610
590 H = H + 1
600 V = INT ( RND (1) * 5):J(T,2
  ) = J(T,2) - H(V,A(X,1),1):J
  (T,4) = J(T,4) - H(V,A(X,1),
  0)
610 NEXT Y,X:M = M1: IF H > 0 THEN
  HTAB 1: VTAB 20: PRINT H" S
  ECONDARY ARMAMENT HITS ON:":
  PRINT J$(T): CALL - 958:M
  = 0
615 FOR ZZ = 1 TO 1111: NEXT ZZ:
  GOTO 880
620 C = C + 1:M = M1 - C: HTAB 1:
  VTAB 17: CALL - 958
630 IF A(7,3) < 1 AND NT < 1 THEN
  PRINT "NO TORPEDOES TO FIRE
  ": FOR ZZ = 1 TO 1111: NEXT
  ZZ: GOTO 880
640 IF A(7,3) < 1 AND NT > 9 THEN
  A(7,3) = 10:NT = NT - 10
650 A(7,3) = A(7,3) - 5: PRINT "T
  ARGET ? (ENTER NUMBER PLEASE
  )":M = M1
670 GET T$:T = VAL (T$): IF T <
  1 OR T > 8 THEN PRINT "NO S
  UCH TARGET": GOTO 670
680 T = T - 1: IF J(T,5) < 1 THEN
  HTAB 1: VTAB 19: PRINT J$(T
  )" SUNK,": PRINT "SELECT AND
  THER TARGET": GOTO 670
690 FOR Y = 1 TO 5:Z = INT (( RND
  (1) * 10) + 1) + M: IF Z > 5
  THEN 720
700 V = INT ( RND (1) * 5):J(T,2
  ) = J(T,2) - H(V,6,1):J(T,4)
  = J(T,4) - H(V,6,0):H = H +
  1
710 HTAB 1: VTAB 19: PRINT H" HI
```

```
TS ON:": PRINT J$(T): CALL
- 958: FOR ZZ = 1 TO 444: NEXT
ZZ
720 NEXT Y: GOTO 880
730 HTAB 1: VTAB 17: CALL - 958
: PRINT " 8 1 2": HTAB 7
: PRINT "!": HTAB 4: PRINT "
7--+-3 ENTER NEW COURSE":
HTAB 7: PRINT "!": HTAB 4: PRINT
"6 5 4"
760 GET I$:G = VAL (I$)
770 IF G < 1 OR G > 8 THEN 760
780 ON G GOTO 790,800,810,820,83
0,840,850,860
790 A1 = - 40: GOTO 870
800 A1 = - 39: GOTO 870
810 A1 = 1: GOTO 870
820 A1 = 41: GOTO 870
830 A1 = 40: GOTO 870
840 A1 = 39: GOTO 870
850 A1 = - 1: GOTO 870
860 A1 = - 41
870 GOTO 190
880 IF C > 1 AND M1 < 3 THEN 102
0
885 M = M1: HTAB 1: VTAB 17: CALL
- 958:H = 0
890 FOR Y = 0 TO 7: IF J(Y,5) <
1 THEN 1010
895 HTAB 1: VTAB 17: PRINT "INCO
MING JAPANESE FIRE FROM:": PRINT
J$(Y): CALL - 958
900 FOR X = 0 TO 7: IF A(X,5) <
1 THEN NEXT X: GOTO 4070
910 T = TT: FOR X = 1 TO 1: NEXT
X:TT = TT + 1: IF TT > 7 THEN
TT = 0
915 IF A(T,5) < 1 THEN 910
920 FOR X = 1 TO J(Y,6):Z = INT
(( RND (1) * 10) + 1) + M: IF
Z > 5 THEN 960
940 M = M + .5:H = H + 1:V = INT
( RND (1) * 5): IF (V = 1 OR
V = 3) AND Y < 4 THEN A(T,6)
= A(T,6) - 1
950 A(T,2) = A(T,2) - H(V,J(Y,0),
1):A(T,4) = A(T,4) - H(V,J(Y
,0),0)
960 NEXT X: IF H > 0 THEN HTAB
1: VTAB 20: PRINT H" MAIN AR
MAMENT HITS ON:": PRINT A$(T
): CALL - 958: FOR ZZ = 1 TO
1111: NEXT ZZ:H = 0
965 IF Y > 5 THEN 1010
970 FOR X = 1 TO J(Y,7) * 2:Z =
INT (( RND (1) * 10) + 1) +
M: IF Z > 5 THEN 1000
980 H = H + 1:V = INT ( RND (1) *
5):A(T,2) = A(T,2) - H(V,J(Y
```

```
,1),1):A(T,4) = A(T,4) - H(V
,J(Y,1),0)
1000 NEXT X: IF H > 0 THEN HTAB
1: VTAB 20: PRINT H" SECONDA
RY ARMAMENT HITS ON:": PRINT
A$(T):H = 0: FOR ZZ = 1 TO 1
111: NEXT ZZ
1010 NEXT Y: FOR X = 1 TO 222: NEXT
X
1015 IF C > 1 THEN 3020
1017 GOTO 160
1020 M = M1: HTAB 1: VTAB 17: CALL
- 958: PRINT "INCOMING JAPA
NESE TORPEDO FIRE": IF M1 =
3 THEN M = M + 1
1025 FOR ZZ = 1 TO 666: NEXT ZZ
1030 FOR X = 4 TO 6: IF J(X,3) >
0 AND J(X,5) > 0 THEN 2040
1035 NEXT X
1040 IF (J(7,3) < 1 AND JT < 1) OR
J(7,5) < 1 THEN 3020
1050 IF J(7,3) < 1 AND JT > 9 THEN
J(7,3) = 10:JT = JT - 10
1060 J(7,3) = J(7,3) - 5
1070 FOR X = 0 TO 7: IF A(X,5) <
1 THEN NEXT X: GOTO 4070
1080 T = TT: FOR X = 1 TO 1: NEXT
X:TT = TT + 1: IF TT > 7 THEN
TT = 0
1085 IF A(T,5) < 1 THEN 1080
1090 FOR X = 1 TO 5:Z = INT (( RND
(1) * 10) + 1) + M: IF Z > 4
THEN 2030
1100 V = INT ( RND (1) * 5):A(T,
2) = A(T,2) - H(V,6,1):A(T,4
) = A(T,4) - H(V,6,0): HTAB
1: VTAB 19: PRINT "TORPEDO H
IT ON "A$(T): CALL - 958: FOR
ZZ = 1 TO 1111: NEXT ZZ: IF
T < 7 THEN T = T + 1
1110 FOR ZZ = 1 TO 1111: NEXT ZZ
2030 NEXT X: GOTO 3020
2040 F = X: FOR X = 1 TO 1: NEXT
X:J(F,3) = J(F,3) - 8
2050 FOR X = 0 TO 7: IF A(X,5) <
1 THEN NEXT X: GOTO 4070
2060 T = X: FOR X = 1 TO 1: NEXT
X: IF T = 8 THEN 4070
2070 FOR X = 1 TO 8:Z = INT (( RND
(1) * 10) + 1) + M: IF Z > 4
THEN 3010
2080 V = INT ( RND (1) * 5):A(T,
2) = A(T,2) - H(V,6,1):A(T,4
) = A(T,4) - H(V,6,0): HTAB
1: VTAB 19: PRINT "TORPEDO H
IT ON "A$(T): CALL - 958: FOR
Z = 1 TO 1111: NEXT Z: IF T <
7 THEN T = T + 1
3010 NEXT X
```

```

3020 HTAB 1: VTAB 17: CALL - 95
      8: PRINT "CHECKING STATUS OF
      ALL SHIPS": FOR ZZ = 1 TO 8
      88: NEXT ZZ
3030 FOR X = 0 TO 6: J(X,7) = 0: IF
      J(X,2) < 1 OR J(X,4) < 1 THEN
      J(X,5) = 0: PRINT J*(X) " SUN
      K",: FOR Z = 1 TO 1111: NEXT
      Z
3050 IF A(X,2) < 1 OR A(X,4) < 1
      THEN A(X,5) = 0
3060 NEXT X: J(7,7) = 0: IF A(7,2
      ) > 0 OR A(7,4) > 0 THEN 309
      0
3070 A(7,5) = A(7,5) - 1: IF A(7,
      5) < 1 THEN 4020
3080 A(7,2) = 5: A(7,4) = 20: A(7,3
      ) = 10: NT = NT - 10
3090 IF J(7,2) > 0 OR J(7,4) > 0
      THEN 4020
4000 J(7,5) = J(7,5) - 1: IF J(7,
      5) < 1 THEN 4020
4010 J(7,2) = 5: J(7,4) = 25: J(7,3
      ) = 10: JT = JT - 10
4020 FOR X = 0 TO 6:: IF J(X,5) <
      1 THEN NEXT X: GOTO 4030
4025 GOTO 4060
4030 HOME : PRINT "ALL JAPANESE

```

```

      CAPITAL SHIPS SUNK": PRINT
4040 PRINT "YOU HAVE SAVED THE L
      EYTE LANDINGS": PRINT : PRINT
4050 PRINT "TO PLAY AGAIN TYPE R
      UN": END
4060 FOR X = 0 TO 7: IF A(X,5) >
      0 THEN 150
4070 NEXT X: HOME : PRINT "YOU L
      OST ALL YOUR SHIPS": PRINT
4080 PRINT "MAYBE KINKAID'S SEVE
      NTH FLEET": PRINT
4090 PRINT "WILL SAVE THE LEYTE
      LANDINGS": PRINT : PRINT : GOTO
      4050
5010 HOME : PRINT "THE JAPANESE
      HAVE REACHED LEYTE": PRINT :
      GOTO 4080
6000 DATA "YAMATO", "NEW JERSEY"
      ,1,2,4,4,35,33,0,0,300,275,1
      ,1,9,9,0,0
6010 DATA "NAGATO", "IOWA", 2,2,4
      ,4,30,33,0,0,250,275,1,1,8,9
      ,0,0
6020 DATA "KONGO", "WASHINGTON",
      3,2,4,5,29,31,0,0,250,250,1,
      1,8,9,0,0
6030 DATA "HAGURO", "MASSACHUSET
      TS", 3,2,4,5,28,31,0,0,250,25

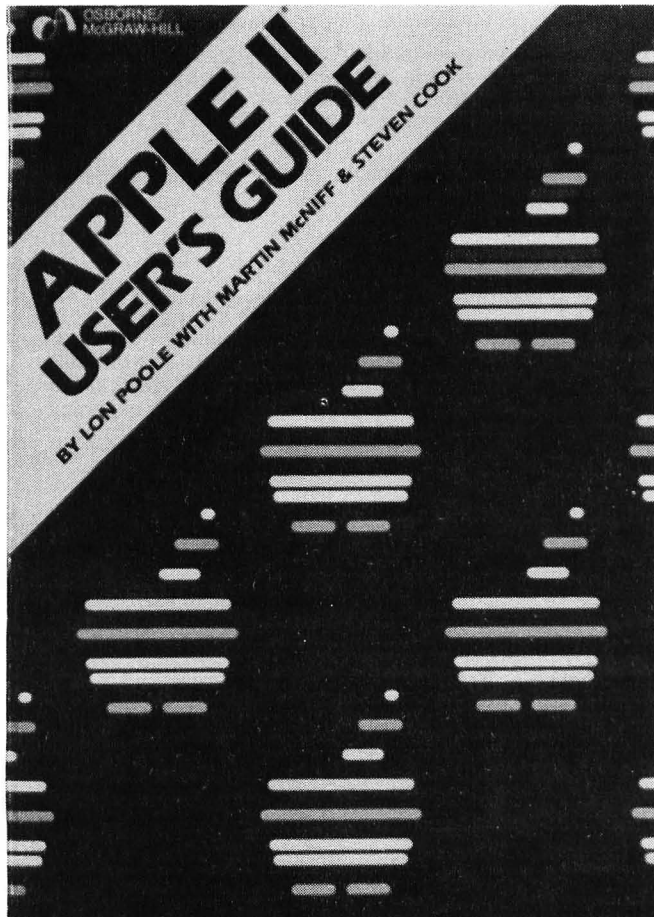
```

```

      0,1,1,8,9,0,0
6040 DATA "CHOKAI", "ALABAMA", 4,
      2,5,5,25,31,16,0,220,250,1,1
      ,8,9,0,0
6050 DATA "SUZUYA", "BALTIMORE",
      4,4,5,5,25,25,16,0,220,200,1
      ,1,15,9,0,0
6060 DATA "NAGARA", "PITTSBURGH"
      ,5,4,0,5,20,25,8,0,100,150,1
      ,1,7,9,0,0
6070 DATA "JAP. DESTROYERS", "U
      .S. DESTROYERS", 5,5,0,5,10,1
      5,10,10,50,55,8,6,6,5,0,0
6080 DATA 20,2,15,1,12,1,8,0,2,
      0,20,1,25,2,20,1,15,1,10,0,3
      ,0,15,1
6090 DATA 18,1,15,1,12,0,7,0,2,
      0,17,3,15,1,15,1,11,0,7,0,2,
      0,17,3
6100 DATA 17,2,15,2,12,2,10,1,3
      ,1,20,5
6110 DATA 361,362,323,284,285,24
      7,208,209,210,211,212,173,17
      4,215,216,177,178,179,180,18
      1,222
6120 DATA 183,224,185,226,227,18
      8,229,270,271,312,313,354,35
      5,396,397,438,479,999,999

```

continued on next page



The Apple II User's Guide

by Lon Poole, Martin McNiff,
and Steven Cook

This guide is the key to unlocking the full power of the Apple* II computer. Topics include: **Applesoft and Integer BASIC programming** — especially how to make the best use of Apple's sound, color, and graphics capabilities. The book presents a thorough description of every BASIC statement, command, and function. **Machine level programming** — although not a Machine Language programming guide, this book covers the Machine Language Monitor in detail. **Hardware features** — the disk drive and printer are covered in separate chapters. **Advanced programming** — special sections describe high resolution graphics techniques and other advanced applications.

*Apple is a trademark of the Apple Computer Corporation

The Apple II User's Guide \$15.00



Ten reasons why your floppy disk should be a BASF FlexyDisk®.



More than four decades of experience in magnetic media – BASF invented magnetic recording tape, the forerunner of today's wide range of magnetic media, back in 1934, and was the first independent manufacturer of IBM-compatible floppy disks.

Tough Tyvek sleeve – no paper dust, no static electricity.

Special self-cleaning jacket and liner help eliminate data errors and media wear and tear.

Packaging to suit your requirements – standard flip-top box, Cassette 10® storage case, or bulk pack.

Center hole diameter punched to more accurate standards than industry specifications, for top performance.

100% certification – every single disk is tested at thresholds 2-3 times higher than system requirements, to be 100% error-free.

Bi-axially oriented polyester substrate – for uniform and reliable performance year after year.

For the name of your nearest supplier, write BASF Systems, Crosby Drive, Bedford, MA 01730, or call 617-271-4030.

Cross-linked oxide coating – for low head wear and long trouble-free media life.

Total capability – one of two manufacturers in the world that makes both 8" and 5.25" models, has tape and disk experience, and manufactures floppy disk drives.

Double lubrication – lubricants both in the formula and on the disk surface, to minimize media wear due to head friction.



BASF

Floppy Disks Mag Cards Cassettes Computer Tapes Disk Packs Computer Peripherals

Atari Version

10 REM ORIGINAL VERSION BY:

20 REM VICTOR A. VERNON JR.

30 REM ATARI VERSION BY:

40 REM ALAN J. ZETT

50 DIM A(7,7),J(7,7),A\$(120),J\$(120),H(4,12),T1\$(40),T2\$(15),R\$(40)

60 GRAPHICS 0:I=PEEK(560)+PEEK(561)*25

6+4:I=PEEK(I)+PEEK(I+1)*256

70 POKE 752,1:?"ONE MOMENT PLEASE":?

"THERE IS A LOT OF DATA TO READ":NT=50

:JT=100:A=167:A1=-1:POKE 82,0

80 FOR X=1 TO 105:J\$(X)=" ":A\$(X)=" ":

NEXT X:RESTORE

100 FOR X=0 TO 7:READ T1\$,T2\$:J\$(X*15+

1)=T1\$:A\$(X*15+1)=T2\$:FOR Y=0 TO 7:REA

D T1,T2:J(X,Y)=T1:A(X,Y)=T2:NEXT Y:NEX

T X

110 FOR X=0 TO 4:FOR Y=1 TO 6:FOR Z=0

TO 1:READ T1:H(X,Y+Z*6)=T1:NEXT Z:NEXT

Y:NEXT X

113 FOR X=1 TO 40:T1\$(X)=CHR\$(160):NEX

T X:OPEN #1,4,0,"K"

115 GRAPHICS 0:POKE 752,1:FOR X=1 TO 1

3:?" ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

,,,,,,":NEXT X

120 POSITION 9,7:?" T1\$(21):POSITION 7

,8:?" T1\$(17):POSITION 5,9:?" T1\$(13):

POSITION 4,10:?" T1\$(11):

125 POSITION 1,11:?" T1\$(6):POSITION 0

,12:?" T1\$(2):POSITION 0,13:PRINT T1\$:

T1\$:

127 POSITION 0,13:?" < SAN BERNARDINO

STR. LEYTE GULF >":

130 POKE I+371,51:POKE I+375,33:POKE I

+379,45:POKE I+383,33:POKE I+387,50

140 POKE 82,2

150 FOR X=1 TO 1:NEXT X:C=0:H=0

160 POSITION 2,16:GOSUB 7000:POKE I+J,

12:POKE I+A,12

165 READ J:IF J=999 THEN 5010

170 M1=3:Z=INT(RND(0)*3)+1:IF Z=1 THEN

J=J+40

172 IF Z=2 THEN J=J-40

175 IF PEEK(I+A+1)<>128 THEN 177

176 POSITION 2,16:?"YOU'RE ABOUT TO R

UN AGROUND!":?"I'LL CORRECT YOUR POSI

TION.":FOR ZZ=0 TO 500:NEXT ZZ:A1=-40

177 IF A+1<0 THEN A1=40

180 A=A+1:POKE I+A,161:POKE I+J,170:IF

A=J THEN POKE I+A,138

190 POSITION 2,16:GOSUB 7000:?"COMMAN

D ?":IF J=A+1 OR J=A-1 OR J=A+40 OR J=

A-40 THEN M1=0

200 POSITION 2,18:?" 1 - STATUS REPOR

T":IF A=J THEN M1=-1

210 ? " 2 - FIRE BUNS":IF A=J+39 OR A=

J-39 OR A=J+41 OR A=J-41 THEN M1=2

220 ? " 3 - FIRE TORPEDOES":IF A=J+1 O

R A=J-1 OR A=J+40 OR A=J-40 THEN M1=0

230 ? " 4 - COURSE CORRECTION":M=M1

240 POKE 764,255:FOR X=1 TO 2000:IF PE

EK(764)=255 THEN 270

250 GET #1,T1:IF T1<49 OR T1>52 THEN 2

70

255 FOR X=1 TO 1:NEXT X

260 G=VAL(CHR\$(T1)):ON G GOTO 280,480,

620,730

270 NEXT X:C=C+1:IF C>2 THEN 150

275 GOTO 240

280 FOR X=0 TO 7:IF A(X,5)<1 THEN R\$="

SUNK":GOTO 430

290 IF X>1 THEN 340

300 IF A(X,4)<50 OR A(X,2)<7 THEN R\$="

FLOATING JUNK YARD":GOTO 430

310 IF A(X,4)<100 OR A(X,2)<12 THEN R\$

="VERY HEAVY DAMAGE":GOTO 430

320 IF A(X,4)<150 OR A(X,2)<17 THEN R\$

="MODERATE DAMAGE":GOTO 430

330 R\$="ESSENTIALLY UNDAMAGED":GOTO 43

0

340 IF X>4 THEN 380

350 IF A(X,4)<50 OR A(X,2)<7 THEN R\$="

PREPARE TO ABANDON SHIP!":GOTO 430

360 IF A(X,4)<100 OR A(X,2)<14 THEN R\$

="SEVERE DAMAGE":GOTO 430

370 R\$="LITTLE OR NO DAMAGE":GOTO 430

380 IF X=4 THEN 420

390 IF A(X,4)<20 OR A(X,2)<4 THEN R\$="

SINKING!":GOTO 430

400 IF A(X,4)<50 OR A(X,2)<8 THEN R\$="

HEAVY DAMAGE!":GOTO 430

410 R\$="LIGHT OR NO DAMAGE":GOTO 430

420 IF A(X,4)<25 OR A(X,2)<5 THEN R\$="

HEAVY DAMAGE!":GOTO 430

425 R\$="UNDAMAGED"

430 POSITION 2,16:GOSUB 7000:?"STATUS

:"

440 ? A\$(X*15+1),(X+1)*15):?" R\$:IF X=7

AND A(7,5)>1 THEN ? A(7,5):" AFLOAT"

460 FOR Y=0 TO 500:NEXT Y:NEXT X

470 GOTO 190

480 C=C+1:FOR X=0 TO 7:IF A(X,5)<1 THE

M NEXT X

490 POSITION 2,16:GOSUB 7000:?"ENTER

TARGET FOR ";A\$(X*15+1),(X+1)*15):M=M1

500 GET #1,T1:T=VAL(CHR\$(T1))

510 IF T<1 OR T>8 THEN POSITION 2,17:?"

"ENTER NUMBER OF TARGET SHIP":?" CHEC

K INSTRUCTIONS":GOTO 500

520 T=T-1:M=M+J(T,7):J(T,7)=J(T,7)-J(T

,7)+0.5

530 IF J(T,5)<1 THEN ? J\$(T*15+1,(T+1)

*15):?" WAS SUNK":FOR ZZ=0 TO 200:NEXT

ZZ:GOTO 490

540 H=0:FOR Y=1 TO A(X,6):Z=INT((RND(0

)*10)+1)+M:IF Z>4 THEN 570

545 H=H+1

550 V=INT(RND(0)*5):IF X<5 AND (V=0 OR

V=1 OR V=3) THEN J(T,6)=J(T,6)-1

555 FOR Z=0 TO 200:NEXT Z

560 J(T,2)=J(T,2)-H(V,A(X,0)+6):J(T,4)

=J(T,4)-H(V,A(X,0))

570 NEXT Y:IF H>0 THEN POSITION 2,18:G

OSUB 7000:?" H;?" MAIN ARMAMENT HITS ON:

":?" J\$(T*15+1,(T+1)*15):FOR ZZ=1 TO 5

580 M=M1:FOR Y=1 TO A(X,6)*2:Z=INT((RN

D(0)*10)+1)+M:IF Z>4 THEN 610

590 H=H+1

600 V=INT(RND(0)*5):J(T,2)=J(T,2)-H(V,

A(X,1)+6):J(T,4)=J(T,4)-H(V,A(X,1))

610 NEXT Y:NEXT X:M=M1:IF H>0 THEN POS

ITION 2,19:GOSUB 7000:?" H;?" SECONDARY

ARMAMENTS HITS ON:":?" J\$(T*15+1,(T+1)*

15):H=0

615 FOR ZZ=0 TO 500:NEXT ZZ:GOTO 880

620 C=C+1:M=M1-C:POSITION 2,16:GOSUB 7

000

630 IF A(7,3)<1 AND NT<1 THEN ? "NO TO

RPEDDES TO FIRE":FOR ZZ=0 TO 500:NEXT

ZZ:GOTO 880

640 IF A(7,3)<1 AND NT>9 THEN A(7,3)=1

0:NT=NT-10

650 A(7,3)=A(7,3)-5:?" TARGET ? (ENTER

NUMBER PLEASE)":M=M1

670 GET #1,T1:T=VAL(CHR\$(T1)):IF T<1 O

R T>8 THEN ? "NO SUCH TARGET":GOTO 670

680 T=T-1:IF J(T,5)<1 THEN POSITION 2,

18:?" J\$(T*15+1,(T+1)*15):?" WAS SUNK":

? "SELECT ANOTHER TARGET":GOTO 670

690 FOR Y=1 TO 5:Z=INT((RND(0)*10)+1)+

M:IF Z>5 THEN 720

700 V=INT(RND(0)*5):J(T,2)=J(T,2)-H(V,

12):J(T,4)=J(T,4)-H(V,6):H=H+1

710 POSITION 2,18:?" H;?" HITS ON: ";J\$(

T*15+1,(T+1)*15):GOSUB 7000:FOR ZZ=0 T

O 200:NEXT ZZ

720 NEXT Y:GOTO 880

730 POSITION 2,16:GOSUB 7000:?" 8

1 2"? " !":?" 7---3 ENTE

R NEW COURSE"

740 ? " !":?" 6 5 4"

760 GET #1,T1:G=VAL(CHR\$(T1))

continued on next page

continued from previous page

```

770 IF G<1 OR G>8 THEN 760
780 GOTO 6*10+780
790 A1=-40:GOTO 870
800 A1=-39:GOTO 870
810 A1=1:GOTO 870
820 A1=41:GOTO 870
830 A1=40:GOTO 870
840 A1=39:GOTO 870
850 A1=-1:GOTO 870
860 A1=-41
870 GOTO 190
880 IF C>1 AND M1<3 THEN 1020
885 M=M1:POSITION 2,16:GOSUB 7000:H=0
890 FOR Y=0 TO 7:IF J(Y,5)<1 THEN 1010
895 POSITION 2,16: "INCOMING JAPANESE
FIRE FROM: " :GOSUB 7000: J*(
Y*15+1,(Y+1)*15)
900 FOR X=0 TO 7:IF A(X,5)<1 THEN NEXT
X:GOTO 4070
910 T=TT:FOR X=1 TO 1:NEXT X:TT=TT+1:IF
TT>7 THEN TT=0
915 IF A(T,5)<1 THEN 910
920 FOR X=1 TO J(Y,6):Z=INT((RND(0)*10
)+1)+M:IF Z>5 THEN 960
940 M=M+0.5:H=H+1:V=INT(RND(0)*5):IF (
V=1 OR V=3) AND Y<4 THEN A(T,6)=A(T,6)
-1
950 A(T,2)=A(T,2)-H(V,J(Y,0)+6):A(T,4)
=A(T,4)-H(V,J(Y,0))
960 NEXT X:IF H>0 THEN POSITION 2,19:
H;" MAIN ARMAMENT HITS ON:" :GOSUB 700
0: A*(T*15+1,(T+1)*15):FOR ZZ=0 TO 50
0:NEXT ZZ:H=0
965 IF Y>5 THEN 1010
970 FOR X=1 TO J(Y,7):Z=INT((RND(0)*10
)+1)+M:IF Z>5 THEN 1000
980 H=H+1:V=INT(RND(0)*5):A(T,2)=A(T,2)
)-H(V,J(Y,1)+6):A(T,4)=A(T,4)-H(V,J(Y,
1))
1000 NEXT X:IF H>0 THEN POSITION 2,19:
? H;" SECONDARY ARMAMENT HITS ON:" : A
*(T*15+1,(T+1)*15):H=0:FOR ZZ=0 TO 500
:NEXT ZZ
1010 NEXT Y:FOR X=0 TO 100:NEXT X
1015 IF C>1 THEN 3020
1017 GOTO 160
1020 M=M1:POSITION 2,16:GOSUB 7000: "

```

```

INCOMING JAPANESE TORPEDO FIRE":IF M1=
3 THEN M=M+1
1025 FOR ZZ=0 TO 300:NEXT ZZ
1030 FOR X=4 TO 6:IF J(X,3)>0 AND J(X,
5)>0 THEN 2040
1035 NEXT X
1040 IF (J(7,3)<1 AND JT<1) OR J(7,5)<
1 THEN 3020
1050 IF J(7,3)<1 AND JT>9 THEN J(7,3)=
10:JT=JT-10
1060 J(7,3)=J(7,3)-5
1070 FOR X=0 TO 7:IF A(X,5)<1 THEN NEX
T X:GOTO 4070
1080 T=TT:FOR X=1 TO 1:NEXT X:TT=TT+1:
IF TT>7 THEN TT=0
1085 IF A(T,5)<1 THEN 1080
1090 FOR X=1 TO 5:Z=INT((RND(0)*10)+1)
+M:IF Z>4 THEN 2030
1100 V=INT(RND(0)*5):A(T,2)=A(T,2)-H(V
,12):A(T,4)=A(T,4)-H(V,6):POSITION 2,1
8
1105 GOSUB 7000: "TORPEDO HIT ON " :A*(
T*15+1,(T+1)*15):FOR ZZ=0 TO 500:NEXT
ZZ:IF T<7 THEN T=T+1
1110 FOR ZZ=0 TO 500:NEXT ZZ
2030 NEXT X:GOTO 3020
2040 F=X:FOR X=1 TO 1:NEXT X:J(F,3)=J(
F,3)-8
2050 FOR X=0 TO 7:IF A(X,5)<1 THEN NEX
T X:GOTO 4070
2060 T=X:FOR X=1 TO 1:NEXT X:IF T=8 TH
EN 4070
2070 FOR X=1 TO 8:Z=INT((RND(0)*10)+1)
+M:IF Z>4 THEN 3010
2080 V=INT(RND(0)*5):A(T,2)=A(T,2)-H(V
,12):A(T,4)=A(T,4)-H(V,6):POSITION 2,1
8
2090 GOSUB 7000: "TORPEDO HIT ON " :A*(
T*15+1,(T+1)*15):FOR ZZ=0 TO 500:NEXT
ZZ:IF T<7 THEN T=T+1
3010 NEXT X
3020 POSITION 2,16:GOSUB 7000: "CHECK
ING FOR SUNKEN SHIPS":FOR ZZ=0 TO 400:
NEXT ZZ
3030 FOR X=0 TO 6:J(X,7)=0:IF J(X,2)<1
OR J(X,4)<1 THEN J(X,5)=0: J*(X*15+1
,(X+1)*15): " ";FOR Z=1 TO 500:NEXT

```

```

Z
3050 IF A(X,2)<1 OR A(X,4)<1 THEN A(X,
5)=0
3060 NEXT X:J(7,7)=0:IF A(7,2)>0 OR A(
7,4)>0 THEN 3090
3070 A(7,5)=A(7,5)-1:IF A(7,5)<1 THEN
4020
3080 A(7,2)=5:A(7,4)=20:A(7,3)=10:NT=N
T-10
3090 IF J(7,2)>0 OR J(7,4)>0 THEN 4020
4010 J(7,2)=5:J(7,4)=25:J(7,3)=10:JT=J
T-10
4020 FOR X=0 TO 6:IF J(X,5)<1 THEN NEX
T X:GOTO 4030
4025 GOTO 4060
4030 GRAPHICS 0: "ALL JAPANESE CAPITO
L SHIPS SUNK":?
4040 ? "YOU HAVE SAVED THE LEYTE LANDI
NGS":? :?
4050 ? "TO PLAY AGAIN TYPE RUN":END
4060 FOR X=0 TO 7:IF A(X,5)>0 THEN 150
4070 NEXT X:GRAPHICS 0: "YOU LOST ALL
YOUR SHIPS":?
4080 ? "MAYBE KINKAID'S SEVENTH FLEET"
:?
4090 ? "WILL SAVE THE LEYTE LANDINGS":
? :? :GOTO 4050
5010 GRAPHICS 0: "THE JAPANESE HAVE R
EACHED LEYTE":? :GOTO 4080
6000 DATA YAMATO,NEW JERSEY,1,2,4,4,35
,33,0,0,300,275,1,1,9,9,0,0
6010 DATA NAGATO,IOWA,2,2,4,4,30,33,0,
0,250,275,1,1,8,9,0,0
6020 DATA KONGO,WASHINGTON,3,2,4,5,29,
31,0,0,250,250,1,1,8,9,0,0
6030 DATA HAGURO,MASSACHUSETTS,3,2,4,5
,28,31,0,0,250,250,1,1,8,9,0,0
6040 DATA CHOKAI,ALABAMA,4,2,5,5,25,31
,16,0,220,250,1,1,8,9,0,0
6050 DATA SUZUYA,BALTIMORE,4,4,5,5,25,
25,16,0,220,200,1,1,15,9,0,0
6060 DATA NAGARA,PITTSBURGH,5,4,0,5,20
,25,8,0,100,150,1,1,7,9,0,0
6070 DATA JAP. DESTROYERS,U.S. DESTROY
ERS,5,5,0,5,10,15,10,10,50,55,8,6,6,5,
0,0
6080 DATA 20,2,15,1,12,1,8,0,2,0,20,1,
25,2,20,1,15,1,10,0,3,0,15,1
6090 DATA 18,1,15,1,12,0,7,0,2,0,17,3,
15,1,15,1,11,0,7,0,2,0,17,3
6100 DATA 17,2,15,2,12,2,10,1,3,1,20,5
6110 DATA 361,362,323,284,285,247,208,
209,210,211,212,173,174,215,216,177,17
8,179,180,181,222
6120 DATA 183,224,185,226,227,188,229,
270,271,312,313,354,355,396,397,438,47
9,999,999
7000 ? " ";RETURN
7010 REM 7 <ESC> <SHIFT> <DELETE>'S

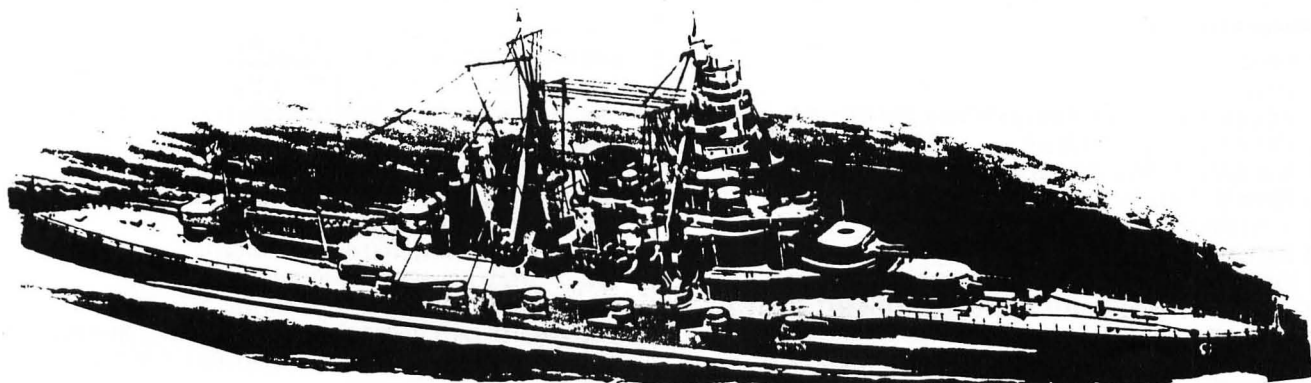
```

Continental Adventures

4975 Brookdale Dept. 01
 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



S-80 Version

```

10 REM LEYTE : BATTLE OFF SAMAR (HYPOTHETICAL) :OCT.,1944
20 REM VICTOR A. VERNON JR.
50 POKE 16553,255
60 CLEAR200:DIMA(7,7),J(7,7),A*(7),J*(7),H(4,6,1):RANDOM
65 ON ERROR GOTO7000
70 CLS:PRINT"ONE MOMENT PLEASE":PRINT"THERE IS A LOT OF DATA TO
READ":NT=50:JT=100:A=265:A1=-1:I=15360
100 FORX=0T07:READJ*(X),A*(X):FORY=0T07:READJ(X,Y),A(X,Y):NEXTY,
X
110 FORX=0T04:FORY=1T06:FORZ=0T01:READH(X,Y,Z):NEXTZ,Y,X
115 FORX=0T0640STEP64:PRINT@X,STRING$(64,46):NEXTX
120 PRINT@394,STRING$(39,191):PRINT@454,STRING$(49,191):PRINT@
513,STRING$(56,191):PRINT@576,STRING$(62,191):PRINT@578,"< SAN
BERNARDINO STR.":
130 PRINT@620,"LEYTE GULF >":POKE15821,83:POKE15828,65:POKE1583
5,77:POKE15842,65:POKE15849,82
135 FORX=15822T015850STEP7:POKEX,32:POKEX-2,32:NEXTX
150 FORX=1T01:NEXTX:C=0:H=0
160 PRINT@640,CHR$(31):POKEI+J,46:POKEI+A,46
165 READJ:IFJ=999THEN5010
170 M1=3:Z=RND(3):IFZ=1THENJ=J+64ELSEIFZ=2THENJ=J-64
175 IFPEEK(I+A+A1)=191PRINT@640,"YOU'RE ABOUT TO RUN AGROUND TUR
KEY":PRINT"THE COMPUTER WILL NOW CORRECT YOUR STUPIDITY":FORZZ=1
T0500:NEXTZZ:A1=-64
176 IFA+A1<0THEHA1=64
180 A=A+A1:POKEI+A,65:POKEI+J,74:IFA=JTHENPOKEI+A,42
190 PRINT@640,"COMMAND ?":CHR$(220):IFJ=A+10RJ=A-10RJ=A+640RJ=A-
64THENM1=0
200 PRINT" 1 - STATUS REPORT":IFA=JTHENM1=-1
210 PRINT" 2 - FIRE GUNS":IFA=J+630RA=J-630RA=J+650RA=J-65THENM1
=2
220 PRINT" 3 - FIRE TORPEDOES":IFA=J+10RA=J-10RA=J+640RA=J-64THE
M1=0
230 PRINT" 4 - COURSE CORRECTION":M=M1
240 FORX=1T02000:I*=INKEY$:IFI$=""THEN270
250 IFASC(I*)<490RASC(I*)>52THEN270ELSEFORX=1T01:NEXTX
260 G=VAL(I*):ONG GOTO280,480,620,730
270 NEXTX:C=C+1:IFC>2THEN150ELSE240
280 FORX=0T07:IFA(X,5)<1THENR$="SUNK":GOTO430
290 IFX>1THEN340
300 IFA(X,4)<500RA(X,2)<7THENR$="FLOATING JUNK YARD":GOTO430
310 IFA(X,4)<1000RA(X,2)<12THENR$="VERY HEAVY DAMAGE":GOTO430
320 IFA(X,4)<1500RA(X,2)<17THENR$="MODERATE DAMAGE":GOTO430
330 R$="ESSENTIALLY UNDAAMAGED":GOTO430
340 IFX>4THEN380
350 IFA(X,4)<500RA(X,2)<7THENR$="PREPARE TO ABANDON SHIP!":GOTO4
30
360 IFA(X,4)<1000RA(X,2)<14THENR$="SEVERE DAMAGE":GOTO430
370 R$="LITTLE OR NO DAMAGE":GOTO430
380 IFX=7THEN420
390 IFA(X,4)<200RA(X,2)<4THENR$="SINKING!":GOTO430

```

```

400 IFA(X,4)<500RA(X,2)<8THENR$="HEAVY DAMAGE!":GOTO430
410 R$="LIGHT OR NO DAMAGE":GOTO430
420 IFA(X,4)<250RA(X,2)<5THENR$="HEAVY DAMAGE!"ELSER$="UNDAAMAGED
"
430 PRINT@640,CHR$(31):"STATUS : "
440 PRINTA*(X):" : ";R$:IFX=7 ANDA(7,5)>1PRINTA(7,5)" AFLOAT"
460 FORY=1T0500:NEXTY:NEXTX
470 GOTO190
480 C=C+1:FORX=0T07:IFA(X,5)<1THENNEXTX
490 PRINT@640,CHR$(31):"ENTER TARGET FOR "A*(X):M=M1
500 T*=INKEY$:IFT$=""THEN500ELSESET=VAL(T*)
510 IFT<10RT>8PRINT@704,"ENTER NUMBER OF TARGET SHIP. CHECK INST
RUCTIONS":GOTO500
520 T=T-1:M=M+J(T,7):J(T,7)=J(T,7)+.5
530 IFJ(T,5)<1PRINTJ*(T)" SUNK":FORZZ=1T0200:NEXTZZ:GOTO490
540 H=0:FORY=1T0A(X,6):Z=RND(10)+M:IFZ>4THEN570ELSEH=H+1
550 V=RND(5)-1:IFX<5AND(V=0ORV=1ORV=3)THENJ(T,6)=J(T,6)-1
555 FORZ=1T0200:NEXTZ
560 J(T,2)=J(T,2)-H(V,A(X,0),1):J(T,4)=J(T,4)-H(V,A(X,0),0)
570 NEXTY:IFH>0PRINT@704,H"MAIN ARMAMENT HITS ON "J*(T):CHR$(31)
:FORZZ=1T0500:NEXTZZ:H=0
580 M=M1:FORY=1T0A(X,6):Z=RND(10)+M:IFZ>4THEN610ELSEH=H+1
600 V=RND(5)-1:J(T,2)=J(T,2)-H(V,A(X,1),1):J(T,4)=J(T,4)-H(V,A(X
,1),0)
610 NEXTY,X:M=M1:IFH>0PRINT@768,H"SECONDARY ARMAMENT HITS ON "J*(
T):CHR$(31):H=0
615 FORZZ=1T0500:NEXTZZ:GOTO880
620 C=C+1:M=M1-C:PRINT@640,CHR$(31)
630 IFA(7,3)<1ANDNT<1PRINT"NO TORPEDOES TO FIRE":FORZZ=1T0500:NE
XTZZ:GOTO880
640 IFA(7,3)<1ANDNT>9THENA(7,3)=10:NT=NT-10
650 A(7,3)=A(7,3)-5:PRINT"TARGET ? (ENTER NUMBER PLEASE)":M=M1
670 T*=INKEY$:IFT$=""THEN670ELSESET=VAL(T*):IFT<10RT>8PRINT"NO SUC
H TARGET":GOTO670
680 T=T-1:IFJ(T,5)<1PRINT@704,J*(T)" SUNK SELECT ANOTHER TARGET"
:GOTO670
690 FORY=1T05:Z=RND(10)+M:IFZ>5THEN720
700 V=RND(5)-1:J(T,2)=J(T,2)-H(V,6,1):J(T,4)=J(T,4)-H(V,6,0):H=H
+1
710 PRINT@704,H"HITS ON "J*(T):CHR$(220):FORZZ=1T0200:NEXTZZ
720 NEXTY:GOTO880
730 PRINT@640,CHR$(31):" 8 1 2":PRINT@710,"!":PRINT@771,"7--
+-3 ENTER NEW COURSE":PRINT@838,"!":PRINT@899,"6 5 4":
760 I*=INKEY$:IFI$=""THEN760ELSESET=VAL(I*)
770 IF<10R>8THEN760
780 ONGGOTO790,800,810,820,830,840,850,860
790 A1=-64:GOTO870
800 A1=-63:GOTO870
810 A1=1:GOTO870
820 A1=65:GOTO870
830 A1=64:GOTO870
840 A1=63:GOTO870
850 A1=-1:GOTO870

```

continued on next page

continued from previous page

```

860 A1=-65
870 B0T0190
880 IFC>1ANDM1<3THEN1020ELSEM=M1:PRINT@640,CHR$(31);:H=0
890 FORV=0T07:IFJ(Y,5)<1THEN1010
895 PRINT@640,"INCOMING JAPANESE FIRE FROM ";J*(Y);CHR$(31)
900 FORX=0T07:IFA(X,5)<1THENNEXTX:GOTO4070
910 T=TT:FORX=1T01:NEXTX:TT=TT+1:IFTT>7THENTT=0
915 IFA(T,5)<1THEN910
920 FORX=1T0J(Y,6):Z=RND(10)+M:IFZ>5THEN960
940 M=M+.5:H=H+1:V=RND(5)-1:IF(V=1ORV=3)ANDY<4THENA(T,6)=A(T,6)-1
950 A(T,2)=A(T,2)-H(V,J(Y,0),1):A(T,4)=A(T,4)-H(V,J(Y,0),0)
960 NEXTX:IFH>0PRINT@768,H"MAIN ARMAMENT HITS ON "A$(T);CHR$(31):FORZZ=1T0500:NEXTZZ:H=0
965 IFY>5THEN1010
970 FORX=1T0J(Y,7)*2:Z=RND(10)+M:IFZ>5THEN1000
980 M=M+1:V=RND(5)-1:A(T,2)=A(T,2)-H(V,J(Y,1),1):A(T,4)=A(T,4)-H(V,J(Y,1),0)
1000 NEXTX:IFH>0THENPRINT@768,H"SECONDARY ARMAMENT HITS ON "A$(T);:H=0:FORZZ=1T0500:NEXTZZ
1010 NEXTX:FORX=1T0100:NEXTX
1015 IFC>1THEN3020ELSE160
1020 M=M1:PRINT@640,CHR$(31);"INCOMING JAPANESE TORPEDO FIRE":IFM1=3THENM=M+1
1025 FORZZ=1T0300:NEXTZZ
1030 FORX=4T06:IFJ(X,3)>0ANDJ(X,5)>0THEN2040ELSENEXTX
1040 IF(J(7,3)<1ANDJT<1)ORJ(7,5)<1THEN3020
1050 IFJ(7,3)<1ANDJT>9THENJ(7,3)=10:JT=JT-10
1060 J(7,3)=J(7,3)-5
1070 FORX=0T07:IFA(X,5)<1THENNEXTX:GOTO4070
1080 T=TT:FORX=1T01:NEXTX:TT=TT+1:IFTT>7THENTT=0

```

```

1085 IFA(T,5)<1THEN1080
1090 FORX=1T05:Z=RND(10)+M:IFZ>4THEN2030
1100 V=RND(5)-1:A(T,2)=A(T,2)-H(V,6,1):A(T,4)=A(T,4)-H(V,6,0):PRINT@704,"TORPEDO HIT ON "A$(T);CHR$(31):FORZZ=1T0500:NEXTZZ:IFT<7THENT=TT+1
1110 FORZZ=1T0500:NEXTZZ
2030 NEXTX:GOTO3020
2040 F=X:FORX=1T01:NEXTX:J(F,3)=J(F,3)-8
2050 FORX=0T07:IFA(X,5)<1THENNEXTX:GOTO4070
2060 T=X:FORX=1T01:NEXTX:IFT=8THEN4070
2070 FORX=1T08:Z=RND(10)+M:IFZ>4THEN3010
2080 V=RND(5)-1:A(T,2)=A(T,2)-H(V,6,1):A(T,4)=A(T,4)-H(V,6,0):PRINT@704,"TORPEDO HIT ON "A$(T);CHR$(31):FORZZ=1T0500:NEXTZZ:IFT<7THENT=TT+1
3010 NEXTX
3020 PRINT@640,CHR$(31);"CHECKING STATUS OF ALL SHIPS":FORZZ=1T0400:NEXTZZ
3030 FORX=0T06:J(X,7)=0:IFJ(X,2)<1ORJ(X,4)<1THENJ(X,5)=0:PRINTJ*(X)"SUNK";:FORZ=1T0500:NEXTZ
3050 IFA(X,2)<1ORJ(X,4)<1THENA(X,5)=0
3060 NEXTX:J(7,7)=0:IFA(7,2)>0ORA(7,4)>0THEN3090
3070 A(7,5)=A(7,5)-1:IFA(7,5)<1THEN4020
3080 A(7,2)=5:A(7,4)=20:A(7,3)=10:NT=NT-10
3090 IFJ(7,2)>0ORJ(7,4)>0THEN4020
4000 J(7,5)=J(7,5)-1:IFJ(7,5)<1THEN4020
4010 J(7,2)=5:J(7,4)=25:J(7,3)=10:JT=JT-10
4020 FORX=0T06:IFJ(X,5)<1THENNEXTXELSEGOTO4060
4030 CLS:PRINTCHR$(23):PRINT"ALL JAPANESE CAPITAL SHIPS SUNK":PRINT
4040 PRINT"YOU HAVE SAVED THE LEYTE":PRINT:PRINT"LANDINGS":PRINT:PRINT
4050 PRINT"TO PLAY AGAIN TYPE RUN":END
4060 FORX=0T07:IFA(X,5)>0THEN150
4070 NEXTX:CLS:PRINTCHR$(23):PRINT"YOU LOST ALL YOUR SHIPS":PRINT

```

```

4080 PRINT"MAYBE KINKAID'S SEVENTH FLEET":PRINT
4090 PRINT"WILL SAVE THE LEYTE LANDINGS":PRINT:PRINT:GOTO4050
5010 CLS:PRINTCHR$(23):PRINT"THE JAPANESE HAVE REACHED LEYTE":PRINT:GOTO4080
6000 DATA "YAMATO","NEW JERSEY",1,2,4,4,35,33,0,0,300,275,1,1,9,9,0,0
6010 DATA "NAGATO","IOWA",2,2,4,4,30,33,0,0,250,275,1,1,8,9,0,0
6020 DATA "KONGO","WASHINGTON",3,2,4,5,29,31,0,0,250,250,1,1,8,9,0,0
6030 DATA "HAGURO","MASSACHUSETTS",3,2,4,5,28,31,0,0,250,250,1,1,8,9,0,0
6040 DATA "CHOKAI","ALABAMA",4,2,5,5,25,31,16,0,220,250,1,1,8,9,0,0
6050 DATA "SUZUYA","BALTIMORE",4,4,5,5,25,25,16,0,220,200,1,1,15,9,0,0
6060 DATA "NAGARA","PITTSBURGH",5,4,0,5,20,25,8,0,100,150,1,1,7,9,0,0
6070 DATA "JAP. DESTROYERS","U.S. DESTROYERS",5,5,0,5,10,15,10,10,50,55,8,6,5,0,0
6080 DATA 20,2,15,1,12,1,8,0,2,0,20,1,25,2,20,1,15,1,10,0,3,0,15,1
6090 DATA 18,1,15,1,12,0,7,0,2,0,17,3,15,1,15,1,11,0,7,0,2,0,17,3
6100 DATA 17,2,15,2,12,2,10,1,3,1,20,5
6110 DATA 384,385,386,387,388,325,326,327,264,265,266,267,268,269,270,207,208,273,274,211,212,213,214,215,216,217,218,283,284,221
6120 DATA 286,287,224,289,290,291,228,229,230,295,296,233,298,299,300,301,301,303,304,305,306,371,372,439,440,441,506,507,508,509
6130 DATA 510,574,999,999
7000 FORX=1T01:NEXTX:FORV=1T01:NEXTV:FORZZ=1T01:NEXTZZ
7100 RESUME150

```

SPORTS FANS!

WORLD SERIES

written by David Bolke

Ah yes, it's springtime 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. Batter up!

S-80 16K Tape \$9.95
 Apple 16K Tape \$9.95
 Apple 32K Disk \$14.95
 Atari 16K Tape \$9.95



MASTERS' GOLF

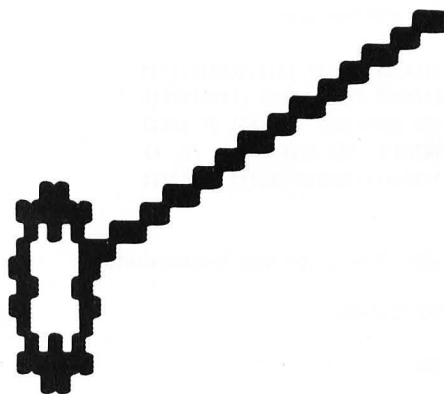
written by David Bolke

Tired of golf as a sport you can only play during the warmer half of the year? Now you can play it year 'round with **MASTERS GOLF!** For the Atari and the S-80, **MASTERS GOLF** promises hours and hours of delight.

S-80 16K Tape \$9.95
 Atari 8K Tape \$9.95



ABM COMMAND



by Arnold E. van Beverhoudt, Jr.

"ABM Command" is an arcade-style game for a 16K S-80.

Enemy missiles are headed for the East Coast! As commander of the U.S. Air Defense Department, you are responsible for the launching of Anti-Ballistic Missiles to destroy incoming ICBMs. Although this simplified BASIC version of the popular arcade game is not as fast-paced as its Machine Language cousins, it is challenging and fun to play.

Your ABM base is centrally located and has 10 missiles for each round. You must move a target marker to direct the course of your ABMs. You score 500 points for each ICBM you destroy, while the occasional spy satellites are worth 1000 points. However, you lose points for each ICBM that hits the ground: 50 points for rural areas between cities, 100 points for outlying Boston and Miami, and 250 points for nearby New York and Washington.

The game normally ends when all four cities have been destroyed; that is, when each has been hit by three ICBMs. But be warned: A direct hit on

your launching base blows the base and your game! For each 25,000 points scored, you receive a bonus city.

The game has five levels of difficulty, which are represented by the size of the warheads in your ABMs. At the easy level, your ABMs are equipped with large warheads; therefore, they don't have to be very accurate. However, at the hardest level your ABM warheads are very small and your aim must be perfect.

Well, those enemy missiles are getting closer. Man your launch command post, and good luck!

VARIABLES:

A,B: Starting coordinates of ICBM flight path.

A\$,B\$,C\$: Graphic strings for title page missiles.

AM: Counter for number of ABMs remaining in each round.

BV: Counter for bonus points.

C,D: Ending coordinates of ICBM flight path.

DL: Difficulty level.

E,F: Starting coordinates of ABM flight path.

G,H: Ending coordinates of ABM flight path.

I,J,K,L: Used to poke sound routine.

IM: Counter for number of ICBM's in each round.

M,N: Coordinates of target marker.

R\$: INKEY\$ cue for next page, etc.

S: USR call for sound.

SS\$: Dummy string to poke sound routine.

T: FOR/NEXT loop counter.

TD: Random direction of spy satellite.

TT: Used in random branch to ICBM or spy satellite routines.

U1,U2,U3,U4: Counters for number of hits on each city.

V: Player's score.

W: Value of PEEK(14400). Used to check for arrow keys.

X,Y: SET/RESET coordinates for ICBM/ABM explosions.

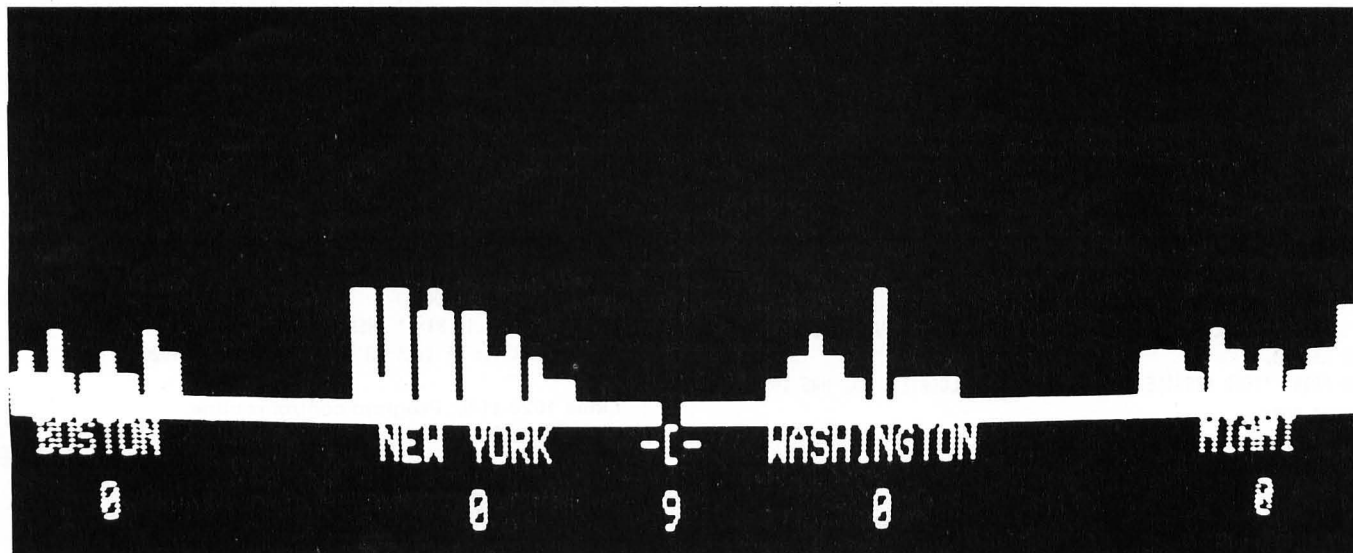
X1,Y1: SET/RESET coordinates for ABM flight path.

X2,Y2: SET/RESET coordinates for ICBM flight path.

X3,Y3: SET/RESET coordinates for spy satellite.

Z: FOR/NEXT loop counter.

continued on next page



continued from previous page

```
10 REM *****
***** ABM COMMAND *****
***** COPYRIGHT (C) 1981 BY *****
** ARNOLD E. VAN BEVERHOUDT, JR. **
*****
```

Lines 120-690: Title page and instructions.

Goto sound routine.

```
120 GOTO 3020
```

Display launching of missiles.

```
130 CLS: FOR T=1T05: PRINT: PRINT: C%=CHR$(191): C1%=CHR$(184):
C2%=CHR$(180): C3%=CHR$(186): C4%=CHR$(181)
140 X=RND(53)+3: A%=C1%+C2%+C3%: B%=C3%+C4%+C1%+C4%
150 PRINT TAB(X+1);A%: PRINT TAB(X);B%: FOR Z=1T06: PRINT TAB(X)
;STRING$(5,C%): NEXT Z
160 PRINT TAB(X+1);"***": PRINT TAB(X);"****": PRINT TAB(X+1);"
***": PRINT TAB(X+2);"***": S=USR(5000): NEXT T
170 FOR T=1T016: PRINT: S=USR(5000): NEXT T
```

Display title page.

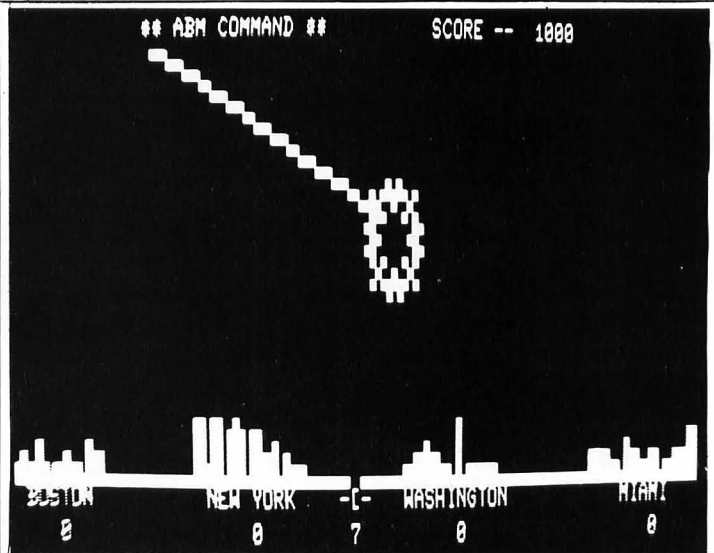
```
180 PRINT CHR$(23): PRINT: PRINT
190 PRINT TAB(10) "ABM COMMAND": PRINT: PRINT: PRINT
200 PRINT TAB(6) "COPYRIGHT (C) 1981 BY": PRINT: PRINT: PRINT
210 PRINT TAB(2) "ARNOLD E. VAN BEVERHOUDT, JR."
220 PRINT: PRINT: PRINT
230 FOR T=1T0750: NEXT T
```

Initialize variables and clear string space.

```
240 CLEAR 500: DEFINT A-U, W-Z: RANDOM
245 X=0: Y=0: G=0: H=0
250 U1=0: U2=0: U3=0: U4=0: V=0: IM=RND(4)+8: AM=10: BV=0
```

Display instructions.

```
260 CLS: PRINT "DO YOU NEED INSTRUCTIONS ('Y' OR 'N') ?"
270 R%=INKEY$: IF R%="" THEN 270ELSE 280
280 IF LEFT$(R%,1)<>"Y" AND LEFT$(R%,1)<>"N" THEN 260ELSE 290
290 IF LEFT$(R%,1)="Y" THEN 300ELSE 600
300 CLS: FOR T=1T07: PRINTTAB(20) "RED ALERT! RED ALERT!": PR
INT
310 S = USR(1): NEXT T: CLS
320 PRINT "THE NORTH AMERICAN AIR DEFENSE COMMAND REPORTS THAT I
NTER-"
330 PRINT "CONTINENTAL BALLISTIC MISSILES ARE HEADED TOWARDS THE
U.S. EAST"
340 PRINT "COAST. AS COMMANDER OF THE U.S. ABM COMMAND, YOU ARE
RESPON-"
350 PRINT "SIBLE FOR GUIDING ANTI-BALLISTIC MISSILES TO INTERCEP
T AND"
360 PRINT "DESTROY THE ICBM'S. YOU DO THIS BY MOVING AN ELECTRO
NIC TARGET"
370 PRINT "MARKER TO INTERCEPT THE ICBM'S FLIGHT PATH AND LAUNCH
ING AN ABM."
380 PRINT "YOUR MISSILE BASE IS CENTRALLY LOCATED, AND HAS 10 MI
SSILES FOR"
390 PRINT "EACH ROUND OF INCOMING ICBM'S.": PRINT
400 PRINT "YOUR ABM BASE IS THE MAIN DEFENSE UNIT FOR BOSTON, NE
W YORK"
410 PRINT "CITY, WASHINGTON, AND MIAMI. BOSTON AND MIAMI ARE FA
R FROM "
```



```
420 PRINT "YOUR ABM BASE, AND ARE HARD TO DEFEND. NEW YORK AND
WASHINGTON"
430 PRINT "HOWEVER, CAN BE EASILY DEFENDED.": PRINT
440 PRINT "PRESS <ENTER> TO CONTINUE.": S=USR(1000)
450 R% = INKEY$: IF R% = "" THEN GOTO 450ELSE 460
460 CLS: PRINT"YOU SCORE 500 POINTS FOR EACH ICBM YOU DESTROY, A
ND 1000 POINTS"
470 PRINT "FOR EACH SPY SATELLITE YOU HIT. BUT, YOUR SCORE IS R
EDUCED BY"
480 PRINT "250 POINTS FOR ICBM HITS ON NEW YORK OR WASHINGTON, 1
00 POINTS"
490 PRINT "FOR HITS ON BOSTON AND MIAMI, AND 50 POINTS FOR HITS
ON RURAL"
500 PRINT "AREAS BETWEEN THE CITIES. IF YOUR ABM BASE IS DESTRO
YED BY A"
510 PRINT "DIRECT ICBM HIT, THE GAME ENDS AND YOU LOSE.": PRINT
520 PRINT "THE GAME NORMALLY ENDS WHEN ALL 4 CITIES HAVE BEEN DE
STROYED,"
530 PRINT "THAT IS, WHEN EACH HAS BEEN HIT BY 3 ICBM'S. FOR EAC
H 25000"
540 PRINT "POINTS YOU SCORE, YOU GET THE BONUS OF HAVING BOSTON
REBUILT.": PRINT
550 PRINT "THE GAME HAS MULTIPLE DIFFICULTY LEVELS, WHICH CONTRO
L THE SIZE"
560 PRINT "OF THE NUCLEAR WARHEAD IN YOUR ABM MISSILES. THE LAR
GER THE"
570 PRINT "WARHEAD, THE EASIER IT IS TO DESTROY THE ICBM'S."
580 PRINT: PRINT "PRESS <ENTER> TO CONTINUE.": S=USR(1000)
590 R% = INKEY$: IF R% = INKEY$ THEN GOTO 590ELSE 600
600 CLS: PRINT TAB(22) "ABM LAUNCH COMMANDS": PRINT
610 PRINT TAB(11) "<ENTER> - TO LAUNCH AN ABM"
620 PRINT TAB(11) CHR$(91);" - TO MOVE TARGET MARKER UP"
630 PRINT TAB(11) CHR$(92);" - TO MOVE TARGET MARKER DOWN"
640 PRINT TAB(11) CHR$(93);" - TO MOVE TARGET MARKER LEFT"
650 PRINT TAB(11) CHR$(94);" - TO MOVE TARGET MARKER RIGHT"
660 PRINT: PRINT "GOOD LUCK! YOU AND THE ENTIRE EAST COAST WILL
NEED IT."
670 PRINT: PRINT "ENTER DIFFICULTY LEVEL - '1' (HARD) TO '5' (
EASY) ?": S=USR(1000)
680 R%=INKEY$: IF R%="" THEN 680ELSE 690
690 DL=VAL(R%): IF DL<1 OR DL>5 THEN 670ELSE 1020
```

Lines 1020-1140: Program control routine.

Reset ICBM and ABM counters for new round.

```
1020 CLS: FOR T=1T0500: NEXT T: GOTO 4020
1030 GOTO 1070
```

```

1040 IM=RND(4)+8: AM=10: PRINT @393, "YOUR SCORE IS ";V;" -- THE
ATTACK CONTINUES"
1050 FOR T=1T010: FOR Z=1000T09000 STEP1000: S=USR(Z): NEXT Z: N
EXT T: PRINT @393,"
"
1060 PRINT @925, "-[-";: PRINT @989, "10";
1070 FOR T=64 TO 704 STEP 64: PRINT @T, STRING$(64,128);: NEXTT

```

Reset target marker to start position.

```

1080 RESET (M,N): M=64: N=20
1090 SET(M,N)
1100 TT=RND(100)

```

Randomly set variable to branch for ICBM or spy satellite.

```

1110 IF TT>=20 THEN GOTO 5020ELSE GOTO 11020

```

Check score for extra city bonus.

```

1120 IF BV>=25000 THEN GOTO 10020ELSE 1130

```

If all 4 cities are destroyed, branch to end-of-game.

```

1130 IF U1>=3 AND U2>=3 AND U3>=3 AND U4>=3 THEN 2020ELSE 1140

```

Test for start of new round.

```

1140 IF IM<1 THEN 1040ELSE 1070

```

Lines 2020-2190: End-of-game routines.

Compute final score and display game results.

```

2020 PRINT @256, STRING$(64,"*");: PRINT, "YOUR FINAL SCORE WAS"
;V;:PRINT:PRINT
2030 IF V<5000 THEN PRINT "ARE YOU AN ENEMY SPY? YOU LET THE E
AST COAST BE DESTROYED": PRINT "AND CAPTURED. YOU WILL BE TRIED
FOR TREASON."
2040 IF V>=5000 AND V<50000 THEN PRINT "WE WON THE WAR, NO THANK
S TO YOU. THE CITIES WERE REDUCED TO": PRINT "RUBBLE, AND THERE
WERE VERY FEW SURVIVORS. YOU WILL BE REDUCED": PRINT "IN RANK
TO 'PRIVATE'."
2050 IF V>=50000 AND V<100000 THEN PRINT "YOU DID A COMMENDABLE
JOB AT DEFENDING THE EAST COAST CITIES.": PRINT "THE ENEMY ATTA
CK WAS REPELLED, BUT CASUALTIES WERE HEAVY. YOU": PRINT "WILL B
E PROMOTED TO COLONEL."
2060 IF V>=100000 THEN PRINT "YOU DID AN OUTSTANDING JOB AT DEFE
NDING THE EAST COAST CITIES.": PRINT "THE CITIES SURVIVED WITH M
INOR DAMAGE AND CASUALTIES.": PRINT "YOU WILL BE RECOMMENDED FOR
THE 'MEDAL OF HONOR'."
2070 PRINT STRING$(64,"*");
2080 FOR T=1T020: FOR Z=1000T09000 STEP 1000: S=USR(Z): NEXTZ: F
ORZ=9000T01000STEP-1000: S=USR(Z): NEXT Z: NEXTT

```

Prompt for new game.

```

2090 CLS: PRINT "DO YOU WANT TO PLAY AGAIN ('Y' OR 'N')";
2100 R$=INKEY$: IF R$="" THEN 2100ELSE 2110
2110 IF LEFT$(R$,1)<>"Y" AND LEFT$(R$,1)<>"N" THEN 2090ELSE 2120
2120 IF LEFT$(R$,1) = "Y" THEN 250ELSE CLS: END
2130 CLS: FOR T=1T016: PRINT STRING$(64,191);: S=USR(2000): NEXT
T
2140 FOR T=1T016: PRINT STRING$(64,128);: S=USR(9000): NEXTT
2150 CLS: PRINT:PRINT:PRINT:PRINT TAB(5) "Y O U J U S T B L
E W I T C O M M A N D E R ! ! !":PRINT
2160 PRINT TAB(5) "Y O U R A B M B A S E W A S D E S T R
O Y E D ! ! !": PRINT

```

```

2170 PRINT TAB(9) "W E L L , S E E Y O U I N S I B E R I
A ."

```

Display comments if ABM base was destroyed.

```

2180 FOR T=1T030: S=USR(25000): NEXT T
2190 GOTO 2090

```

Lines 3020-3090: Initialize sound.

```

3020 SS$="////////////////////////////////////"
3030 I = VARPTR(SS$): J = PEEK(I+1)+256*PEEK(I+2)
3040 FOR K=JTOJ+26: READ L: POKE K,L: NEXT K
3050 IF PEEK(16396)=201 POKE 16526,PEEK(I+1): POKE 16527,PEEK(I+
2): GOTO 3070
3060 CMD"T": DEFUSRO=PEEK(I+1)+256*PEEK(I+2): POKE 14308,0
3070 DATA 205,127,10,77,68,62,1,105,211,255,45,32,253,60
3080 DATA 105,211,255,45,32,253,13,16,238,175,211,255,201
3090 GOTO 130

```

Lines 4020-4140: Display cities.

```

4020 PRINT@720,CHR$(176)+CHR$(160)+CHR$(144)
4030 PRINT@770, CHR$(144)+STRING$(3,128)+CHR$(160);
4040 PRINT@784, CHR$(191)+CHR$(170)+CHR$(149)+CHR$(190)+CHR$(148
)+CHR$(188)+CHR$(128)+CHR$(144);
4050 PRINT@805, CHR$(144)+STRING$(2,128)+CHR$(149)+STRING$(15,12
8)+CHR$(160)+STRING$(6,128)+CHR$(188);
4060 PRINT@832, CHR$(190)+CHR$(188)+CHR$(189)+CHR$(184)+CHR$(190
)+CHR$(188)+CHR$(186)+CHR$(191);
4070 PRINT STRING$(8,176)+CHR$(191)+CHR$(190)+CHR$(181)+CHR$(191
)+CHR$(181)+STRING$(2,191)+CHR$(181)+CHR$(189)+CHR$(188);
4080 PRINT STRING$(4,176)+CHR$(128)+STRING$(4,176)+CHR$(188)+STR
ING$(2,191)+CHR$(189)+CHR$(180)+CHR$(181)+STRING$(3,188);
4090 PRINT STRING$(9,176)+STRING$(2,191)+CHR$(188)+CHR$(186)+CHR
$(191)+CHR$(188)+CHR$(191)+CHR$(184)+CHR$(190)+STRING$(2,191);
4100 PRINT @ 897, "BOSTON          NEW YORK  -";CHR$(91);"-
WASHINGTON          MIAMI"
4110 PRINT @ 11, "## ABM COMMAND ##          SCORE --";V;
4120 PRINT @989, AM;
4130 GOSUB 8280
4140 GOTO 1030

```

Lines 5020-5150: ICBM flight path.

Randomly set ICBM start and end points.

```

5020 A=RND(127): B=3: C=RND(119)+4: D=36: IM=IM-1

```

Set position of ICBM. Check for key press.

```

5030 IF IM<1 THEN 1120
5040 IF ABS(B-D) > ABS(A-C) THEN GOTO 5100
5050 FOR X2=ATOC STEP SGN(C-A)
5060 Y2=B+(B-D)/(A-C)*(X2-A)
5070 IF Y2>35 THEN GOTO 8020ELSE GOTO 5080
5080 SET(X2,Y2): S=USR(500): GOTO 6020
5090 NEXT X2: GOTO 5150
5100 FOR Y2=BTOD STEP SGN(D-B)
5110 X2=A+(Y2-B)*(C-A)/(D-B)
5120 IF Y2>35 THEN GOTO 8020ELSE GOTO 5130
5130 SET(X2,Y2): S=USR(500): GOTO 6020
5140 NEXT Y2
5150 CLS: GOTO 1120

```

Lines 6020-6360: Inkey and target marker routines.

Peek memory for key press....

continued on next page

continued from previous page

```
6020 W=PEEK(14400)
6030 IF W=16GOTO6150
6040 IF W=8 GOTO 6170
6050 IF W=64 GOTO 6190
6060 IF W=16 GOTO 6210
6070 IF W=32 GOTO 6230
6080 IF W=40 GOTO 6250
6090 IF W=72 GOTO 6280
6100 IF W=48 GOTO 6310
6110 IF W=80 GOTO 6340
```

Branch to ABM launch routine.

```
6120 IF TT<20 THEN GOTO 6130ELSE GOTO 6140
6130 IF TD=1 THEN GOTO 11070ELSE GOTO 11120
6140 IF ABS(B-D)>ABS(A-C) THEN GOTO 5140ELSE 5090
6150 E=60: F=40: G=M: H=N: AM=AM-1
6160 IF AM>=0 THEN 7020ELSE S=USR(9000): PRINT @925, "OUT";
NT @989, " 0 "; GOTO 6120
```

Set new target marker position.

```
6170 RESET(M,N): N=N-1: IF N<6 THEN N=6
6180 SET (M,N): GOTO 6120
6190 RESET(M,N): M=M+2: IF M>123 THEN M=123
6200 SET(M,N): GOTO 6120
6210 RESET(M,N): N=N+1: IF N>30 THEN N=30
6220 SET(M,N): GOTO 6120
6230 RESET(M,N): M=M-2: IF M<4 THEN M=4
6240 SET(M,N): GOTO 6120
6250 RESET(M,N): M=M-1: N=N-1: IF M<4 THEN M=4
6260 IF N<6 THEN N=6
6270 SET(M,N): GOTO 6120
6280 RESET(M,N): M=M+1: N=N-1: IF M>123 THEN M=123
6290 IF N<6 THEN N=6
6300 SET(M,N): GOTO 6120
6310 RESET(M,N): M=M-1: N=N+1: IF M<4 THEN M=4
6320 IF N>30 THEN N=30
6330 SET(M,N): GOTO 6120
6340 RESET(M,N): M=M+1: N=N+1: IF M>123 THEN M=123
6350 IF N>30 THEN N=30
6360 SET(M,N): GOTO 6120
```

Lines 7020-7160: ABM flight path.

Set and move ABM position. Branch to ABM explosion.

```
7020 IF ABS(F-H) > ABS(E-G) THEN GOTO 7100
7030 FOR X1=ETOG STEP SGN(G-E)
7040 Y1=F+(F-H)/(E-G)*(X1-E)
7050 SET(X1,Y1): S=USR(1000): IF Y1>35 THEN RESET(X1,Y1)
7060 NEXT X1: GOTO 9020
7070 IF TT<20 THEN 7080ELSE 7090
7080 IF TD=1 THEN GOTO 11070ELSE GOTO 11120
7090 IF ABS(B-D)>ABS(A-C) THEN GOTO 5140ELSE 5090
7100 FOR Y1=FTOH STEP SGN(H-F)
7110 X1=E+(Y1-F)*(G-E)/(H-F)
7120 SET(X1,Y1): S=USR(1000): IF Y1>35 THEN RESET(X1,Y1)
7130 NEXT Y1: GOTO 9020
7140 IF TT<20 THEN 7150ELSE 7160
7150 IF TD=1 THEN GOTO 11070ELSE GOTO 11120
7160 IF ABS(B-D)>ABS(A-C) THEN GOTO 5140ELSE 5090
```

Lines 8020-8330: ICBM explosion.

Display explosion.

```
8020 FOR T=0T02
```

```
8030 X=C-T-1: Y=D: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
8040 X=C: Y=D-T-1: SET(X,Y):SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1,
Y-1): SET(X-1,Y+1)
8050 X=C+T+1: Y=D: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
8060 X=C: Y=D+T+1: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
8070 X=C-T: Y=D-T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
8080 X=C+T: Y=D-T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
8090 X=C+T: Y=D+T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
8100 X=C-T: Y=D+T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
8110 FOR T1=1000T01005: S=USR(T1): NEXT T1
8120 NEXT T: FOR T=0T02
8130 X=C-T-1: Y=D: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8140 X=C: Y=D-T-1: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8150 X=C+T+1: Y=D: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8160 X=C: Y=D+T+1: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8170 X=C-T: Y=D-T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8180 X=C+T: Y=D+T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8190 X=C+T: Y=D+T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8200 X=C-T: Y=D+T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
8210 FOR T1=1000T01005: S=USR(T1): NEXT T1: NEXT T
```

Decrement score for ICBM hits.

```
8220 IF C=60 OR C=61 THEN GOTO 2130ELSE 8230
8230 IF C<21 THEN V=V-100: BV=BV-100: U1=U1+1: GOTO 8280ELSE GO
TO 8240
8240 IF C>26 AND C<57 THEN V=V-250: BV=BV-250: U2=U2+1: GOTO 828
0ELSE GOTO 8250
8250 IF C>64 AND C<93 THEN V=V-250: BV=BV-250: U3=U3+1: GOTO 828
0ELSEGOTO 8260
8260 IF C>100 THEN V=V-100: BV=BV-100: U4=U4+1: GOTO 8280ELSE GO
TO 8270
```

Count ICBM hits on cities.

```
8270 V=V-50: BV=BV-50: GOTO 8280
8280 IF U1<3 THEN PRINT@963,U1; ELSE PRINT@960,"DESTROYED";
8290 IF U2<3 THEN PRINT@980,U2; ELSE PRINT@977,"DESTROYED";
8300 IF U3<3 THEN PRINT@999,U3; ELSE PRINT@995,"DESTROYED";
8310 IF U4<3 THEN PRINT@1018,U4; ELSE PRINT@1014,"DESTROYED";
8320 PRINT @47, V;
8330 FOR T=1T0250: NEXTT: GOTO 1120
```

Lines 9020-9370: ABM explosion.

Display explosion.

```
9020 IF DL=1 THEN 9030ELSE 9090
9030 X=B: Y=H
9040 SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1,Y-1): SET(X-1
,Y+1)
9050 FOR T1=300T0500 STEP10: S=USR(T1): NEXT T1
```



```

9060 RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): RESET(X+1,Y-1):
  RESET(X-1,Y+1)
9070 FOR T1=3000TO500 STEP 10: S=USR(T1): NEXT T1
9080 GOTO 9290
9090 FOR T=0TODL-2
9100 X=G-T-1: Y=H: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
9110 X=G: Y=H-T-1: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
9120 X=G+T+1: Y=H: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
9130 X=G: Y=H+T+1: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
9140 X=G-T: Y=H-T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
9150 X=G+T: Y=H-T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+
1,Y-1): SET(X-1,Y+1)
9160 X=G+T: Y=H+T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
9170 X=G-T: Y=H+T: SET(X,Y): SET(X+1,Y+1): SET(X-1,Y-1): SET(X+1
,Y-1): SET(X-1,Y+1)
9180 FOR T1=3000TO500STEP25: S=USR(T1): NEXT T1
9190 NEXT T: FOR T=0TODL-2
9200 X=G-T-1: Y=H: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9210 X=G: Y=H-T-1: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9220 X=G+T+1: Y=H: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9230 X=G: Y=H+T+1: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9240 X=G-T: Y=H-T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9250 X=G+T: Y=H-T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9260 X=G+T: Y=H+T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9270 X=G-T: Y=H+T: RESET(X,Y): RESET(X+1,Y+1): RESET(X-1,Y-1): R
ESET(X+1,Y-1): RESET(X-1,Y+1)
9280 FOR T1=3000TO500STEP25: S=USR(T1): NEXT T1: NEXT T

```

Check if ABM HIT ICBM or spy satellite, and update score.

```

9290 PRINT @989, AM;
9300 IF TT<20 THEN GOTO 9330ELSE 9310
9310 IF POINT(X2,Y2)=-1 THEN SET(M,N): GOTO 9360ELSE GOTO 9320
9320 IF POINT(X2,Y2)=0 THEN V=V+500: BV=BV+500: PRINT @ 47, V;
  GOTO 9350
9330 IF POINT(X3,10)=-1 THEN SET(M,N): GOTO 9370ELSE GOTO 9340
9340 IF POINT(X3,10)=0 THEN V=V+1000: BV=BV+1000: PRINT @47, V;
  GOTO 9350
9350 FOR T=1TO250: NEXTT: GOTO 1120
9360 IF ABS(F-H)>ABS(E-B) THEN GOTO 7160ELSE 7090
9370 IF TD=1 THEN GOTO 11070ELSE GOTO 11120

```

Lines 10020-10070: Display bonus city.

```

10020 PRINT @ 399, "BONUS CITY -- BOSTON HAS BEEN REBUILT";
10030 FOR T=1TO10: FOR Z=9000TO1000 STEP-1000: S=USR(Z): NEXT Z:
  NEXT T: PRINT @399, " ";
10040 U1=0: PRINT @963, U1;
10050 PRINT @770, CHR$(144)+STRING$(3,128)+CHR$(160);

```

```

10060 PRINT @832, CHR$(190)+CHR$(188)+CHR$(189)+CHR$(184)+CHR$(1
90)+CHR$(188)+CHR$(186)+CHR$(191);
10070 BV=0: GOTO 1130

```

Lines 11020-11130: Spy satellite routine.

Select random direction.

```

11020 TD=RND(2)
11030 ON TD GOTO 11040, 11090

```

Set motion across screen. Check for key press.

```

11040 FOR X3=1TO126 STEP RND(5)+2: S=USR(9000)
11050 SET(X3,10): SET(X3-1,9): SET(X3-1,11): SET(X3+1,9): SET(X3
+1,11)
11060 GOTO 6020
11070 RESET(X3,10): RESET(X3-1,9): RESET(X3-1,11): RESET(X3+1,9)
: RESET(X3+1,11)
11080 NEXT X3: GOTO 1120
11090 FOR X3=126TO1 STEP-RND(5)+(-2): S=USR(9000)
11100 SET(X3,10): SET(X3-1,9): SET(X3-1,11): SET(X3+1,9): SET(X3
+1,11)
11110 GOTO 6020
11120 RESET(X3,10): RESET(X3-1,9): RESET(X3-1,11): RESET(X3+1,9)
: RESET(X3+1,11)
11130 NEXT X3: GOTO 1120

```



END ZONE II

by Roger W. Robitaille, Sr.

Rough and tumble gridiron action, from the toss of the coin to the 2-minute warning... Four 15-minute quarters, provisions for interceptions, touchbacks, timeouts, fumbles, penalties. Everything except the cheerleaders.

S-80 16K Tape
..... \$14.95

S-80 32K Disk
..... \$15.95



**SoftSide
Selections**
6 South Street Milford NH 03055



Developing Data Base

The DATABASE Series is copyrighted 1981 by Mark Pelczarski. It may be reprinted with written permission from the author.

Since the last "Database" article appeared in **SoftSide**, I've had plenty of requests to have the program listed in its entirety, along with a compiled set of instructions for using the database. This article has the Apple RAM version listed (with several modifications from the last published copy) and a summary of the options. I hope to have a new Atari version soon, so stay tuned. I'd also like to hear from anyone interested in doing some translations for the TRS-80. The next step is to use random-access disk files for storage, and the Apple version is already well ahead of that. Anyone interested in advanced updates should let me know. For now, here's the final RAM version for the Apple!

Running The Database

The first choice you will be presented is whether to initialize a new file or load an existing file. The first time you use it, you'll have to initialize a file. Thereafter, when you want to access that data, you will load the file. Anytime you want to create a new file with a different type of data, you'll use the initialize option. Several different files will fit on a disk and you can use as many different data disks as you like. A few examples of files would be a mailing list (name, address, city, etc.), checkbook list (to whom, withdrawals, deposits...), and an inventory list (stock number, description number, in stock, on order, etc.). Whatever records you want to keep can usually be stored in this type of database format.

To initialize a new file, you must give your file a name, then tell the computer how many headings you want and the names of the headings. An example would be a file named "Addresses," with six headings: Name, Street Address, City, State, Zip Code, and Phone Number. You might want to add an extra heading (or more) for some kind of code. I might use "Computer" for my seventh heading, so I would know what kind of computer a particular person owns. (Not necessary for your average Christmas card list....)

Your data will be organized into what you can picture as a table. The

headings should be your column headings, and each row would have one set of information across those headings. A set of such information is called one record. Once a file has been created, any future time that you use the database you only have to give the file name ("Addresses" is our example) and all the information will automatically be loaded from disk.

The Main Options

After initialization or loading, you are given a list of choices for manipulating your database. Here are the choices in brief form:

- (S) SAVE current data
- (P) PRINT data (to screen or printer)
- (A) ADD new data
- (C) CHANGE some of your data (such as an address change)
- (D) DELETE a record (no card for him next year!)
- (T) SORT
- (F) FILE names - catalogs the disk
- (N) NEW data file - equivalent to quitting and re-running program)
- (Q) QUIT - done (don't use RESET to get out or you may lose some data)

Adding A Record

This is your logical first choice since, with no data in memory, the other options aren't too much fun. Choose (A) from the options page and you'll be asked for information to fill each of your headings for one record. After you've filled one record, you'll be returned to the options page. Note that commas and semicolons don't work in the data. Also see the note below about searching and sorting numeric fields, if you plan to do such.

Printing A Record

To see if your data is really there, type "P" to print your record. The program will ask if you want it put in a special format (S) or default format (D). Choose (D) for the moment. After choosing, you'll be asked if you want it on the screen (S) or the printer (P). Then, after that choice, a list of headings will be displayed, followed by the choices "Begin" and "Return to Menu." Choose the number next to the word "Begin" and press RETURN. Each record that you have in memory will be displayed in sequence. If you're

printing them to the screen, pressing any key advances to the next record. The ESC key returns you to the option page. All the other choices mentioned above will be explained under "searching" and "formatting."

Searching

When printing, changing, or deleting records, you have the choice of selecting individual items, subsets of your data, or the entire set of data. This is done through the search routine. When you used the print routine above, you chose to print all the data by selecting "Begin" before any other choice. Each of the headings is also listed at that point, along with "Record Number." By choosing the number beside any of the headings or "record number," you elect to do a search under that heading. You are then asked if you want to look for an item that is less than or equal, equal, or greater than or equal, to a value you'll give. After choosing 1, 2, or 3, respectively, you'll be asked for a value for comparison. Example: If you want to search for all records with names starting with A through G, you would want NAME, <, G, where "G" is the value used for comparison. If you want all records from number 20 through the end of the file, you would choose RECORD NUMBER, >, 20.

You also have the option of specifying the beginning of a value for comparison. If you wanted all records from people whose zip code starts with a "60" (as 60185), you can specify ZIP CODE, =, 60*. The asterisk says that anything may follow. This is also an easy way to find records without knowing exact information. If you can't spell "Pelczarski" (or if you don't like typing) you can try "PEL*" and you'll find the record.

To start the actual search, you must choose "Begin." A hidden option here is you can specify several search criteria. I might, for example, want to find everyone in my list whose zip code starts with a "60" and who owns an Apple. I would specify ZIP CODE, =, 60*, then I would specify COMPUTER, =, APPLE, and then tell it to begin. The program will ask if I want the item to meet **all** of the conditions, or **any** of the conditions.

"ALL" would find only those with zip 60 who also own an Apple. "ANY" would find everyone with zip 60, plus everyone owning an Apple (technically, everyone with zip 60, or owning an Apple). Up to eight such criteria may be specified, so you may look for everyone whose name starts with D through F (>D and <F), zip starts with 9, and owns an Atari and an Apple, etc.

Changing Records

To change a record, choose (C) from the options page. After specifying whatever search criteria you want, the appropriate record(s) will be shown on the screen. The items under each heading will then be shown in sequence, and the program will wait for you to type "K" to keep, "C" to change, or "R" if the remainder of the record is okay. If you type "C", you'll be asked for the information with which to replace the old item.

Deleting Records

After choosing (D) from the options page and going through the search steps, the record(s) in question will be displayed, and you'll be asked to verify that you want one particular record deleted. Once it's deleted, it's gone. Type "Y" to delete.

Saving A File

When you want to sign off for the day (or even for a minute), typing (S) from the options page saves your current file on disk. It's an excellent idea, especially with important information, to do this twice, using two separate disks, one as a backup. Just in case...

Sorting

The (T) option from the menu allows your items to be sorted in ascending or descending order, under any heading. Alphabetic items are sorted alphabetically; and numeric items are sorted as strings. The latter means that numbers don't always sort the way you want. 125, 34, and 7 will come out in that order due to the first character in each. To get a true numeric sort for now, you'll have to add leading zeros to the maximum number of places, such as 007, 034, and 125 which will then force proper sequencing.

Catalog

From the program, you can get a catalog directly from the options page by typing (F) for filenames. This is very helpful if you can't remember a second file name or, more importantly, a format name.

Switching Data Files

You can load or create a new file

without rerunning the program by selecting (N) from the options. Be sure to verify that the current file has been saved.

Formatting Output

The formatter in this database may be one of the most versatile ones around. Although there are still a few things it can't do, it does a lot that most "professional" databases don't allow. Basically, you can specify the exact form in which you want each record printed. Each record is printed in sequence, meaning that you cannot mix records across a page. At the present time, there is no way to print a one-time heading. (Hint for the adventurous programmer: A one-time heading has to be done before line 3050, which jumps to the search routine.) You can specify which headings are to be printed and where, which items are to be printed and where, and what (if any) additional character strings should be printed on the form. (You may want to include your company name, an expanded version of a heading instead of the heading itself, or just some lines to separate items.)

To create a format, choose the special format option when printing. You'll be asked if you want to load or

continued on next page

Paper Tractor™

The Paper Tractor™ turns your tractor feed printer into a friction feed printing system. Inexpensively.

With The Paper Tractor you can abandon dull continuous form stock and move into new realms of high quality personalized writing paper. Without costly hardware modifications.

This revolutionary product acts as a carrier, protecting and guiding any standard letter or legal size paper through the printing area. Any paper. Bond, parchment, onionskin, vellum, NCR, manuscript paper (sheet music). Even your company or personal letterhead.

The Paper Tractor is a unique solution to the restrictive nature of the tractor feed printer.

The Paper Tractor is designed to work with:

- Matrix, Impact, Daisywheel and Ink Ejection printheads.
- Front, back and dual tractor paper feeds.
- Forward and reverse plotting printers.



The Paper Tractor is also designed with:

- Self-aligning feed.
- Automatic positioning of letter or legal size paper with an edge-protecting guide.

The Paper Tractor has been:

- Proven with all major manufacturer's printers, including Epson, Tiger, TI, GP-80M, DEC-Writer, Commodore, Centronics, Radio Shack V VI VII, Beehive, Vista, Grafix-Plus, Axiom, Coment, HI-Q, DIP, C-ITOH, Cromemco, Okidata and many more.

#15-266001S \$19.50

TSE-HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

continued from previous page

create one. The first time, you'll have to create it. Draw out exactly what you want printed for your form. You'll be telling the computer, line by line, what it looks like. Your choices are (1) Heading, (2) Item, (3) Tab, (4) Next line, (5) String, and (6) End. Here's one example using the "ADDRESSES" file I mentioned earlier. The format will print mailing labels like this:

Mark Pelczarski
1206 Kings Circle
West Chicago IL
60185

Here are the format commands (numerically, my headings are 1 Name, 2 Address, 3 City, 4 State, 5 Zip, 6 Phone, 7 Computer):

Commands	What to Type
Item, Name	2,1
Next Line, 1	4,1
Item, Address	2,2
Next Line, 1	4,1
Item, City	2,3
Tab, 16	3,16
Item, State	2,4
Next Line, 1	4,1
Tab, 12	3,12
Item, Zip	2,5
Next Line, 3	4,3
End	6

The "1" after the next line means to skip down one line. The "3" at the end skips down three lines before printing the next label. Note that none of the actual headings are used in this format, and neither is the phone number.

Another example is a format that will print a separate little form for each person in the database. For lack of a better example, I'll have the following printed:

```

!
-----
THE FOLLOWING PERSON OWNS
AN

                APPLE

NAME JOE TATE
PHONE 555-1212
-----

```

Here's the format to do it:

```

String, -----
Next Line, 1
String, THE FOLLOWING PERSON
OWNS AN
Next Line, 2
Tab, 9

```

```

Item, Computer
Next Line, 2
Heading, Name (Type "1" for hdg.#)
Tab, 7
Item, Name (Type "1" for item #)
Next Line, 1
Heading, Phone (hdg.#6)
Tab, 7
Item, Phone
End

```

I'll let the top line of the next item to be printed be the bottom line for the last, so I can just end the format after printing the last item.

That's all there is to formatting. Play around with it a little to see what it does for you. After a format is created, you'll be asked to name it, and it will automatically be saved to disk. In the future, you'll be able to load it back in when you need it.

A Few Final Words

There are a lot of things this database program still cannot do, but it is a good introduction for those of you who don't know all of what a database program can be used for. As yet, it has no real numeric capability; it doesn't take advantage of of disk capabilities; and some of the routines are slowwww. On the plus side, it has a lot of the features you should look for in a database, and if you ever decide to shop for one (they're expensive) you'll have an idea of the features to consider. I'm amazed that there are \$200 database programs out there that don't even have basic sorting functions; and most only have a rather primitive print formatting.

Thanks to those who have contributed suggestions to this program; many have been incorporated into it already, and dozens are still working their way in. It may be of interest that we are using the random access disk version of the Apple database in our business (recently renamed "Penguin Software"...funny how those things start) and we're keeping our mailing lists, inventory lists, registration card lists, and even an author/article list for a book on it. In a way it's good, because as the program gets bigger, it takes longer to test for bugs. On the other hand, when we find one...zap!

In answer to the many (many, many, many...) requests Mark has received for help on typos, he is making available a disk copy of the Apple Database for \$5 (including postage and handling). Write to Penguin Software, 1206 Kings Circle, West Chicago, IL, 60185, and be sure to state you're requesting the Apple version. SoftSide takes no responsibility for products offered by other companies.

```

10 REM DEVELOPING DATA BASE
15 REM COPYRIGHT 1981
20 REM MARK PELCZARSKI
100 D# = CHR#(4)
101 MX = 200
105 DIM C$(7),C1$(7),C2$(7),F$(5)
)
110 HOME : PRINT "(I) INITIALIZE
    A NEW DATA SET"
120 PRINT "(L) LOAD A PREVIOUSLY
    SAVED DATA SET ?";
130 GET A#; PRINT A#
140 IF A# = "L" THEN GOSUB 1000
    : GOTO 200
150 IF A# = "I" THEN GOSUB 1500
    : GOTO 200
160 GOTO 130
200 POKE 216,0; HOME : PRINT "(S
    ) SAVE CURRENT DATA"
220 PRINT "(P) PRINT DATA"
230 PRINT "(A) ADD DATA"
240 PRINT "(C) CHANGE A RECORD"
250 PRINT "(D) DELETE A RECORD"
260 PRINT "(T) SORT"
270 PRINT "(F) FILE NAMES"
280 PRINT "(N) NEW DATA FILE"
290 PRINT "(Q) QUIT"
295 PRINT : PRINT "YOU HAVE ROOM
    FOR ", FRE (0) - 50, " MORE
    CHARACTERS"
300 GET A#; PRINT A#; PRINT
320 IF A# = "S" THEN GOSUB 2000
    : GOTO 200
330 IF A# = "P" THEN GOSUB 3000
    : GOTO 200
340 IF A# = "A" THEN GOSUB 4000
    : GOTO 200
350 IF A# = "C" THEN SB = 3; GOSUB
    8000; GOTO 200
360 IF A# = "D" THEN SB = 4; FS =
    1; GOSUB 8000; GOTO 200
370 IF A# = "T" THEN GOSUB 7000
    : GOTO 200
380 IF A# = "F" THEN GOSUB 600;
    GOTO 200
400 IF A# = "Q" OR A# = "N" THEN
    500
410 GOTO 200
500 IF SB = 1 THEN 540
510 PRINT "CURRENT FILE IS NOT S
    AVED."; PRINT "DO YOU STILL
    WANT TO QUIT? (Y/N) "; GET
    T#
520 IF T# = "N" THEN 200
530 IF T# < > "Y" THEN 510
540 IF A# = "N" THEN CLEAR : GOTO
    100
550 END
600 PRINT D#;"CATALOG"; GET A#; RETURN

```

Load subroutine.

```
1000 INPUT "FILE NAME? ";F#
1010 ONERR GOTO 1310
1020 PRINT D#;"OPEN";F# + ".DAT"
1030 PRINT D#;"READ";F# + ".DAT"
```

```
1040 INPUT NH; INPUT NI
1130 DIM H$(NH),I$(NX,NH)
1140 FOR I = 0 TO NH: INPUT H$(I)
); NEXT
1200 IF NI = - 1 THEN 1280
1240 FOR I = 0 TO NI
1250 FOR J = 0 TO NH
1260 INPUT I$(I,J)
1270 NEXT J: NEXT I
1280 PRINT D#;"CLOSE";F# + ".DAT"
```

```
1300 SS = 1: RETURN
1310 PRINT "FILE NOT FOUND": GET
A#: POKE 216,0: GOTO 110
```

Initialize subroutine.

```
1500 INPUT "GIVE YOUR FILE A NAME
E : ";F#
1510 IF F# = "" THEN 1500
1520 INPUT "HOW MANY HEADINGS? "
;NH
1530 IF NH < 1 THEN 1520
1540 NH = NH - 1:NI = - 1
1560 DIM H$(NH),I$(NX,NH)
1570 FOR I = 0 TO NH
1580 PRINT "HEADING #";I + 1; INPUT
" : ";H$(I)
1590 NEXT I
1600 SS = 0: RETURN
```

Write subroutine.

```
2000 PRINT "USE ";F#;" AS NAME (
Y/N)?" : GET A#: PRINT A#
2050 IF A# = "Y" THEN 2090
2060 IF A# < > "N" THEN 2000
2070 INPUT "NAME? ";F#
2080 IF F# = "" THEN 2070
2090 ONERR GOTO 2290
2100 PRINT D#;"OPEN";F# + ".DAT"

2110 PRINT D#;"WRITE";F# + ".DAT"

2120 PRINT NH: PRINT NI
2130 FOR I = 0 TO NH
2140 PRINT H$(I)
2150 NEXT
2220 IF NI = - 1 THEN 2270
2230 FOR I = 0 TO NI
2240 FOR J = 0 TO NH
2250 PRINT I$(I,J)
```

```
2260 NEXT J: NEXT I
2270 PRINT D#;"CLOSE";F# + ".DAT"
"
2280 SS = 1: RETURN
2290 PRINT "DISK ERROR": GET A#:
GOTO 200
```

Print subroutine.

```
3000 IF NI = - 1 THEN GOSUB 90
00: RETURN
3005 PRINT "(S) SELECT FORMAT, D
R (D) DEFAULT": GET A#: PRINT
3006 IF A# = "S" THEN GOSUB 100
00:FS = 2: GOTO 3010
3007 IF A# < > "D" THEN 3005
3008 FB = 1
3010 PRINT "(S) SCREEN, OR (P) P
RINTER": GET A#: PRINT
3020 IF A# = "P" THEN SB = 2: GOTO
3050
3030 IF A# < > "S" THEN 3010
3040 SB = 1: PRINT : PRINT "AFTER
EACH RECORD (ESC) WILL RETU
RN TO": PRINT "THE MENU, ANY
OTHER KEY CONTINUES."
3050 PRINT "<PRESS ANY KEY>": GET
A#: GOSUB 8010
3090 IF SB = 2 THEN PRINT D#;"P
R#0"
3100 RETURN
```

Print one record to screen.

```
3300 HOME : ON FS GOSUB 3700,380
0
3310 IF SB = 2 THEN 3350
3340 IF SB < > 4 THEN GET A#: IF
A# = CHR$(27) THEN RS = 1
3350 RETURN
```

Print one default.

```
3700 PRINT : PRINT "RECORD ";I +
1: PRINT
3710 FOR J = 0 TO NH
3720 PRINT H$(J),I$(I,J)
3730 NEXT J
3740 RETURN
```

Print one format.

```
3800 J = 1:T = 0:B# = ""
3820 J1 = VAL ( MID$( F$(T),J,1)
);J = J + 1
3830 IF J1 < 5 THEN N = VAL ( MID$(
F$(T),J,2));J = J - 2
3840 ON J1 GOTO 3850,3860,3870,3
890,3910,3970
```

```
3850 A# = H$(N): GOTO 3950
3860 A# = I$(I,N): GOTO 3950
3870 B# = LEFT$( B#,N - 1): IF LEN
(B#) < N - 1 THEN FOR J2 =
LEN (B#) TO N - 2:B# = B# +
" "; NEXT
3880 GOTO 3960
3890 PRINT B#: IF N > 1 THEN FOR
J2 = 2 TO N: PRINT : NEXT
3900 B# = "": GOTO 3960
3910 IF J > LEN (F$(T)) THEN T =
T + 1:J = 1
3920 J2 = J
3930 IF MID$( F$(T),J2,1) < >
"! " THEN J2 = J2 + 1: GOTO 3
930
3940 A# = MID$( F$(T),J,J2 - J):
J = J2 + 1
3950 B# = B# + A#
3960 IF J > LEN (F$(T)) THEN T =
T + 1:J = 1
3965 GOTO 3820
3970 PRINT B#: RETURN
```

Add subroutine.

```
4000 SS = 0:NI = NI + 1
4005 PRINT : PRINT "RECORD ";NI +
1: PRINT
4010 FOR J = 0 TO NH
4020 PRINT H$(J); INPUT " : ";I
$(NI,J)
4030 NEXT J
4040 RETURN
```

Change subroutine.

```
5000 PRINT : PRINT "(C) CHANGE I
TEM, (K) KEEP ITEM, OR": PRINT
"(R) KEEP REMAINDER OF RECOR
D"
5030 PRINT : PRINT "RECORD ";I +
1
5040 CS = 1:RS = 0: FOR J = 0 TO
NH
5050 PRINT : PRINT H$(J); " : ";I
$(I,J); " "
5055 IF RS = 1 THEN PRINT : GOTO
5090
5060 GET A#: IF A# < > "C" AND
A# < > "K" AND A# < > "R" THEN
5060
5070 PRINT A#: IF A# = "K" THEN
5090
5075 IF A# = "R" THEN RS = 1: GOTO
5090
5080 PRINT H$(J); INPUT " : ";I
$(I,J)
5085 CS = 0
5090 NEXT J
```

continued on next page

continued from previous page

```
5095 RS = 0
5100 IF CS = 0 THEN SS = 0
5110 RETURN
```

Delete subroutine.

```
6000 PRINT : PRINT "DELETE THIS
RECORD? ";
6070 GET A$: IF A$ < > "Y" AND
A$ < > "N" THEN 6070
6080 PRINT A$: IF A$ = "N" THEN
6150
6100 FOR I1 = I + 1 TO NI
6110 FOR J = 0 TO NH
6120 I$(I1 - 1, J) = I$(I1, J)
6130 NEXT J: NEXT I1
6135 FOR J = 0 TO NH: I$(NI, J) =
": NEXT
6140 NI = NI - 1: SS = 0: I = I - 1
6150 RETURN
```

Sort subroutine.

```
7000 IF NI = - 1 THEN GOSUB 90
00: RETURN
7010 PRINT : FOR J = 0 TO NH
7020 PRINT "(, J + 1, )": H$(J)
7030 NEXT J
7040 INPUT "SORT ON WHICH HEADIN
G?": J1
7045 J1 = J1 - 1
7050 IF J1 < 0 OR J1 > NH THEN RETURN
7060 PRINT "(A) ASCENDING, OR (D
) DESCENDING": GET A$
7070 IF A$ = "A" THEN A = 1: GOTO
7100
7080 IF A$ = "D" THEN A = 2: GOTO
7100
7090 GOTO 7060
7100 FOR I = 0 TO NI - 1
7110 T = I
7120 FOR I1 = T + 1 TO NI
7122 PRINT I: "": I1
7125 ON A GOTO 7130, 7140
7130 IF I$(I1, J1) < I$(T, J1) THEN
T = I1
7135 GOTO 7145
7140 IF I$(I1, J1) > I$(T, J1) THEN
T = I1
7145 NEXT I1
7150 IF T = I THEN 7180
7155 FOR J = 0 TO NH
7160 T$ = I$(T, J): I$(T, J) = I$(I,
J): I$(I, J) = T$
7170 NEXT J
7180 NEXT I
7200 SS = 0: RETURN
```

Search subroutine.

```
8000 IF NI = - 1 THEN GOSUB 90
00: RETURN
8010 I1 = 0: I2 = NI: J = 0: C1%(0) =
- 1: BS = 1
8015 HOME : PRINT "SEARCH CRITER
IA:": PRINT
8020 PRINT "0) RECORD NUMBER"
8030 FOR I = 0 TO NH: PRINT I +
1: "": H$(I): NEXT I
8035 PRINT : PRINT NH + 2: "): BEG
IN"
8036 PRINT NH + 3: "): RETURN TO M
ENU"
8040 VTAB 21: INPUT "SELECT : ":
I: IF I < 0 OR I > NH + 3 THEN
8040
8045 IF I = NH + 2 THEN C1%(J) =
- 1: GOTO 8150
8046 IF I = NH + 3 THEN RETURN
```

```
8050 C1%(J) = I - 1
8060 VTAB 22: PRINT "(1) SMALLER
(2) EQUAL (3) LARGER ": GET
A$: PRINT A$: IF A$ < "1" OR
A$ > "3" THEN 8060
8070 C2%(J) = VAL (A$)
8080 VTAB 23: PRINT "COMPARED TO
": J: IF C1%(J) = - 1 THEN
8100
8090 INPUT "": C$(J): J = J + 1: IF
J > 7 THEN 8160
8095 GOTO 8015
8100 INPUT "": I: IF I < 1 OR I >
NI + 1 THEN 8100
8105 I = I - 1
8110 IF C2%(J) = 1 THEN I2 = I
8120 IF C2%(J) = 2 THEN I1 = I: I
2 = I
8130 IF C2%(J) = 3 THEN I1 = I
8140 GOTO 8015
8150 IF J < 2 THEN 8200
8160 VTAB 22: PRINT "1) ITEM MUS
T MEET ALL CONDITIONS": PRINT
"2) ITEM MAY MEET ANY CONDIT
ION": GET A$: IF A$ < "1" OR
A$ > "2" THEN 8160
8170 BS = VAL (A$)
8200 RS = 0: IF BS = 2 THEN PRINT
D$: "PR#1"
8250 I = I1 - 1: FOR I3 = I1 TO I
2: I = I + 1
8255 AS = 0: FOR J = 0 TO 7
8260 IF C1%(J) = - 1 THEN J = 7
: GOTO 8345
8270 ON C2%(J) GOTO 8280, 8290, 83
10
8280 IF I$(I, C1%(J)) < = C$(J) THEN
8330
8285 GOTO 8340
```

```
8290 IF I$(I, C1%(J)) = C$(J) THEN
8330
8295 IF RIGHT$(C$(J), 1) < > "
#" THEN 8340
8298 T = LEN (C$(J)) - 1: IF LEN
(I$(I, C1%(J))) < T THEN 8340
8302 IF LEFT$(I$(I, C1%(J)), T) =
LEFT$(C$(J), T) THEN 8330
8305 GOTO 8340
8310 IF I$(I, C1%(J)) > = C$(J) THEN
8330
8320 GOTO 8340
8330 IF BS = 2 THEN AS = 1: J = 7
8335 GOTO 8345
8340 IF BS = 1 THEN AS = 2: J = 7
8345 NEXT J
8350 IF AS = 0 AND BS = 1 THEN 8
355
8352 IF AS < > 1 THEN 8380
8355 GOSUB 3300
8365 IF SB = 3 THEN GOSUB 5000
8370 IF SB = 4 THEN GOSUB 6000
8375 IF RS = 1 THEN I3 = I2
8380 NEXT I3
8390 PRINT : PRINT "THAT'S ALL":
GET A$: PRINT
8400 RETURN
```

Error subroutine #1

```
9000 PRINT "THERE'S NO DATA IN M
ENORY."
9010 FOR I = 1 TO 1000: NEXT : RETURN
```

Print formatting.

```
10000 IF F$(0) = "" THEN 10040
10010 PRINT "SAME FORMAT?": GET
A$: PRINT
10020 IF A$ = "Y" THEN RETURN
10030 IF A$ < > "N" THEN 10010
10040 PRINT "(L) LOAD FORMAT, OR
(C) CREATE FORMAT": GET A$
: PRINT
10050 IF A$ = "C" THEN 10200
10060 IF A$ < > "L" THEN 10040
10090 ONERR GOTO 10170
10100 INPUT "FORMAT NAME:": A$
10110 PRINT D$: "OPEN": A$ + ".FMT
"
10120 PRINT D$: "READ": A$ + ".FMT
"
10130 INPUT NF
10140 FOR J = 0 TO NF: INPUT F$(
J): NEXT
```

```

10150 PRINT D$;"CLOSE";A$ + ".FM
T"
10160 RETURN
10170 PRINT "FORMAT NOT FOUND"; GET
A$; GOTO 200
10200 NF = 0; J = 0; F$(0) = ""
10210 HOME ; PRINT "START IN THE
UPPER LEFT CORNER AND WORK
ACROSS EACH LINE."
10220 PRINT "1;HEADING, 2;ITEM,
3;TAB, 4;NEXT LINE, 5;STRIN
G, 6;END"; INPUT J1
10230 IF J1 < 1 OR J1 > 6 THEN 1
0220
10240 F$(NF) = F$(NF) + STR$(J1
); J = J + 1
10250 ON J1 GOTO 10260,10260,103
00,10300,10350,10400
10260 FOR T = 0 TO NH; PRINT T +
1;); "H$(T); NEXT
10270 INPUT "WHICH?"; T; T = T - 1
; IF T < 0 OR T > NH THEN 10
270
10280 GOTO 10310
10300 INPUT "HOW MANY?"; T; IF T <
1 OR T > 99 THEN PRINT "OUT
OF RANGE."; GOTO 10300
10310 A$ = STR$(T); IF T < 10 THEN
A$ = "0" + A$
10320 F$(NF) = F$(NF) + A$; J = J +
2
10330 GOTO 10380
10350 INPUT "STRING"; A$; A$ = A$
+ "!";
10360 IF LEN (A$) + J > 255 THEN
NF = NF + 1; J = 0; F$(NF) = ""
10370 F$(NF) = F$(NF) + A$; J = J +
LEN (A$)
10380 IF J > 252 THEN NF = NF +
1; J = 0; F$(NF) = ""
10390 GOTO 10220
10400 INPUT "FORMAT NAME"; A$
10405 ONERR GOTO 10460
10410 PRINT D$;"OPEN";A$ + ".FMT
"
10420 PRINT D$;"WRITE";A$ + ".FM
T"
10430 PRINT NF; FOR J = 0 TO NF;
PRINT F$(J); NEXT
10440 PRINT D$;"CLOSE";A$ + ".FM
T"
10450 RETURN
10460 PRINT "DISK ERROR"; GET A$
; B0SUB 10400

```

Take a big byte of the Apple...

Apple II + Computer



48K RAM
 (#07-247003H) \$1299.00

...and keep nibbling at the edges with these
 peripherals, available from TSE-HARDSIDE...

SUP-R-MOD RF Modulator (#09-223015H)	\$34.95
APPLE II Disk Controller w/Drive (#07-247001H).....	\$619.00
APPLE II Disk Drive (#07-247002H)	\$499.00
MICROSOFT Z-80 SoftCard (#07-207010H)	\$269.00
MICROSOFT RAMCard (#07-207011H)	\$159.00
PASCAL Language Card (#07-247004H)	\$459.00
MOUNTAIN COMPUTER Apple Clock (#07-253001H).....	\$269.00
MOUNTAIN COMPUTER ROMWriter (#07-253004H).....	\$169.00
MOUNTAIN COMPUTER ROMPlus+ w/Filter (#07-253003H).....	\$189.00
MOUNTAIN COMPUTER Music System (#07-253005H).....	\$519.00
MOUNTAIN COMPUTER A/D + D/A (#07-253006H).....	\$329.00
MOUNTAIN COMPUTER Expansion Chasis (#07-253007H) ..	\$609.00
HARDSIDE 16K Memory Upgrade Kit (#09-245002H).....	\$39.00
APPLE Silentype Printer w/Interface (#07-247005H)	\$569.00
Parallel Printer Card w/Cable (#07-223044H).....	\$100.00
CCS Asynchronous Serial Interface (#07-249002H)	\$159.95
CCS Calendar/Clock Module (#07-249001H)	\$125.00
LYNX Communications Interface (#07-226004H).....	\$299.00
VERSAWRITER (#07-248001H)	\$239.00
ALF AM-II (#07-248002H)	\$189.00
NEC 12" Monochrome Monitor (#09-223043H).....	\$239.00
AMDEK Color Monitor (#09-223043H)	\$399.00

TERMS: Prices and specifications are subject to change. TSE HARD SIDE accepts VISA & MASTERCARD, Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). TSE HARD SIDE pays all shipping charges (within the 48 states) on all PREPAID orders over \$100.00. On all orders under \$100 a \$2.50 handling charge must be added. COD orders accepted (orders over \$250 require 25% deposit), there is a \$5.00 handling charge. UPS Blue Label and Air Freight available at extra cost.


TSE-HARDSIDE
 6 South St., Milford, NH 03055 (603) 673-5144
 TOLL FREE OUT-OF-STATE 1-800-258-1790

WE MIGHT BE GOOD FOR EACH OTHER

SoftSide Publications is constantly on the lookout for those with the special skills necessary to enhance our growing family of publications. A leader in the field of BASIC software publishing for the Apple™, Atari™ and TRS-80™, **SoftSide** is currently expanding its search to include editors, machine language programmers, researchers, librarians and marketing people. The creative environment at **SoftSide** is such that multi-skilled people have every opportunity offered to them for career fulfillment. Computer literacy is a clear advantage (though not a requirement) as is a grasp of communication techniques. **SoftSide** Publications speak to the technical end of microcomputer entertainment, its creation and innovation. We provide the surroundings for people to grow as far as they want. All they have to have is the motivation.

Although New Hampshire wages are not the highest available, we offer a better scale of pay than most of our competitors and have a goal-oriented bonus program. We have excellent fringe benefits and New Hampshire has no personal income or sales taxes.

Milford, New Hampshire, is in the beautiful southern area of the state, only an hour's drive from Boston, the Atlantic Ocean and the White Mountains. The state's two largest cities, Manchester and Nashua are only fifteen minutes away. Ski slopes abound in the region, as do fine restaurants, arts and crafts centers and the beautiful countryside of New England.

If you are a skilled person with experience in the fields of computers, entertainment or magazines, and if you enjoy a relaxed country lifestyle with the convenience and cultural opportunities of major urban centers nearby, let us know. **We might be good for each other!** A formal resume isn't necessary, just send a letter outlining your skills, qualifications and experience to:

Randal L. Kottwitz
SoftSide Publications
6 South Street
Milford, NH 03055





I-\$STRING PRODUCES ENVYRN™

by Roger W. Robitaille Sr.

I-\$string refers to a special concept utilizing advance codifying techniques to compact ...? Well — so much for the definitions. Actually, I-\$string was conceived to suggest Intelligent (use of) String. However, the name itself isn't very good. Where possible, a name should suggest what it is about.

Envyrn™ is a derivation of "environ," as in "environment." That is what **Envyrn™** does. It permits you to model an environment.

Envyrn™ should be considered in similar terms to word processors, **Visicalc™**, and database managers. It's a broad-based tool. **Scriptsit™**, **Supertext II™**, and **Word Pro™** are all examples of word processors. They manipulate words and paragraphs, make corrections, customize letters, number pages, print manuscripts, and set type. They are human tools to control and maximize use of one of mankind's greatest inventions — language, or should I say its symbolic equivalent — letters forming words.

Number manipulators, like **Visicalc™** and **Column Calculator™**, are prepared to handle another of mankind's great inventions — numbers — symbolic numerical values. These number manipulators extend our control over numeric information. In the past, numbers were largely used to store results and avoid recalculation. These number manipulators have encouraged the use of numbers in a much more dynamic way. The "what if's?" have become feasible questions. In the end, the user starts to change his view of problems to include application of the capabilities of his new tool.

Database managers like **Soft-Side's™**, **CCA Data Management System™**, and **Data Factory™** handle another class of information. Data files are, in essence, related fact groupings, relating: How tall? How heavy? What blood type? How much? When?, etc.. Whatever the proper labels, they're related to each other in a very specific manner — what databases are all about. The problem relates to information control, and the tool extends the ability to search and compare, analyze, selectively list, act on, etc.. While each information's meaning is very special, the useful manipulations are essentially the same.

You will note that they all have something in common. They each control and manipulate a type of human

information. The system of letters relating to sounds, and groups of letters aping sound formations, substantially preceded word processors. Letters and written words always represented a tool that extended and stored language through symbols. Word processors can merely take it a step further. Index cards were quite an invention, in their day. They represent an obvious recognition of the significance of data and the desire to manipulate it efficiently. Database managers can take the next step. Storing numeric information is not exactly new, either. Using past trends to predict future expectations is how accurate navigation must have started. Precision came with hard bought experience. The numeric modules controlled by the number manipulators clearly invite the next step. Refining the problem, rather than simplifying it, may now be the principle consideration. With the computer accomplishing hours of calculations in minutes, without a complaint, numeric modules may be constructed to yield invaluable conclusions. Every one of these programming concepts points to the manipulation of a certain class of information. **Envyrn™** is yet another class of manipulator — a visual one.

A Visual Manipulator

There are many forms of information which may be stored as visual images: blueprints, maps, charts, even game boards. **Envyrn™** invites a dynamic approach to visual information. "Position" is acknowledged as the core piece of information for many problems and problem approaches. **Envyrn™** offers several approaches for graphically presenting information and allows the "position" to take on characteristics meaningful to the problem.

Envyrn™ is focused on spatial problems. Road maps are an excellent example of sophisticated sets of recurring patterns and symbols. Look at the legend on any map and appreciate the quantity of information organized on a spatial plane. The particular configuration of a game board, like **Easy Money™**, is unique and meaningful, as is an electronic diagram. In each case, the spatial relationship of the elements themselves is a crucial feature of the information being modeled.

Envyrn™ defines a small group of

meaningful and flexible pictures, including some control information. It creates a string array to hold the identification of which picture (and characteristics) to be used in what position. Once the graphic pictures and characteristics join the arrayed data, the **Envyrn™** project proceeds to become a utility. The world mushrooms and grows smart.

The Tiles

The graphic structure of **Envyrn™** is crucial to most of its applications. That structure's closest artistic cousins are design tiles such as floor and ceramic wall tiles. Designs are accomplished by arranging large numbers of smaller pieces.

Due to this similarity, we borrow the term "tile" to describe any single graphic design prepared in support of the map array. One of the pages in the poster pullout bears a complete legend of the tiles used in this month's database. There are actually three different but similar tiles prepared for each map code. These are three different sizes of the same idea.

Variety of Applications

Envyrn™ has as many practical applications as it does entertainment applications. Learning to use **Envyrn™** will be no overnight task. Anyone persistent enough to get through the initiation process will be able to manipulate **Envyrn™**. However, effectively utilizing it will be an acquired skill. For some, **Envyrn™** is the program they've been awaiting to solve the problem that's especially theirs.

One of **Envyrn's™** special strengths is its amusement applications. The same utility can prepare and manipulate an excellent three-dimensional "Star Trek" game, be able to create a role-playing game like an adventure or "Hellfire Warrior," and still help you plan your garden. These features seem certain to attract the creative energies of a bumper crop of computer novelists, far out of proportion to the possible financial gain. Our obligation is to constantly improve **Envyrn's™** usefulness and flexibility — a fate we gladly accept.

Envyrn™ is a very important piece of software. Not only is it not finished, it has only just begun. If **Envyrn™** were to be treated as a product to be sold, it wouldn't be ready for several years — machine coded (it's presently

BASIC), shiny-skinned and expensive. The **Envyrn**TM editor contained in these pages is nothing more than a prototype. The theme editor we are working on now is definitely in another league. At that, this prototype possesses some powerful capabilities that can already be put to use. It is in this light that we are publishing the code.

As with any quality piece of software, **Envyrn**TM has so many subtle applications that practice is required before problems may be thought of with regard to its capabilities. The conceptual practice that this prototype offers is worthwhile in itself. It was this editor which produced the map in the centerfold. It was also with this editor that we laid out the floorplans of **SoftSide**'s offices, including placement of all our furnishings and equipment. Electronic design and musical scoring can be done — and maps — **Envyrn**TM loves map applications such as floor plans, road maps, and zoning charts. You can design some of the more unique alphabets found in the world and print banners utilizing them. **Envyrn**TM becomes a superior banner

maker if you own one of the graphics printers like the Microlines or the MX-80. The editor and the **GAMEPLAY** program are all you need to easily prepare graphic adventures, heretofore only dreams.

Speed

Both the prototype editor published in these pages and the **GAMEPLAY** Interpreter on DV suffer from the slow execution of BASIC. We recognize the problem and have every intention to transfer the concept to another language with greater speed. Lance Micklus has agreed to conduct the translation to Z80 machine code once the logic of the program is more established.

Mr. Ray Cote, former editor of **BYTE**, is now part of the project. His specific assignment is to translate the concept to the Apple II. This first effort for the Apple II will also be under BASIC. In due time, all code will be object machine code or its compiled equivalent.

In order to appreciate the rather

unusual way **Envyrn**TM is being brought to the market, you should understand that **Envyrn**TM is at least the precursor to a whole new generation of software where more time is devoted to the problem than the code. **Envyrn**TM is only a tool. Like other tools it must be used to be appreciated. The more it is used, the better the user becomes, and he will use it all the more. The same can be said of your favorite word processor, database manager, and of course, **VisiCalc**TM.

Unlike these other tools, the applications produced under **Envyrn**TM will frequently be useful to many more individuals than the originator. Its abilities in the entertainment area are most noticeable. However, the applications don't end there. Many of life's problems are essentially spatial in nature — the wiring of a house, the planning of a garden. Even electronic designs may be prepared using **Envyrn**TM.

SoftSide has never printed programs more significant than the ones contained in this issue. We truly expect the concepts behind **Envyrn**TM are going to stir the software community.



About The Centerfold...

No doubt you've noticed an extra wrinkle in your magazine this month — a centerfold. We realize that Hugh Hefner is safe from our competition. The reason behind this addition is quite simple. To understand **Envyrn**TM, you must appreciate the significance of the area it controls. The centerfold itself is only one ninth the size of the original printout. A Microline 83 was used with a fresh ribbon. For those of you into specifications, try these on for size. The main area is encoded into an array tightly controlled to simulate a grid of 99 elements by 99 elements. That's 9,801 positions — the size of six full-sized sheets of quarter-inch graph paper. Each of these positions can point to a menu with 256 selections. This particular application is focused on its tiled graphics capabilities. Because this is a preliminary editor and

because we weren't sure how much space was left on DV, the **Envyrment**TM this represents is really pointless as anything other than a showcase of some of **Envyrn**'sTM capabilities. There wasn't time to write and refine a plot. Another interesting statistic is the fine resolution. This poster is a paste-up of a printout that was forty pages long. The implied resolution of this approach is over one million pixels. With all of its complexity, this map required less than sixty different tiles.

Within the sixty or so defined map codes, there are several families of graphics, such as the shoreline set, the road set, and the wall and door set, which had to be coordinated so they would blend when grouped together to form the larger effects of buildings and road networks. There is a definite skill to con-

ceiving and executing effective tile designs. Practice. Imagine, if a proper set of symbols can be prepared, some of your problems may be on the verge being solved.

There is one special effect used in this month's database. To fully appreciate the effect, the database should be run under **GAMEPLAY/BAS**. **GAMEPLAY/BAS** is an **Envyrn**TM interpreter being prepared for **Diversions thru Envyrn**TM magazine. Like the editor, it too is at the prototype stage. However, the jump feature is in effect, permitting preprogrammed jumps in map position tied to specific places. Thus, when certain places are reached, the effect of a scene change takes place with the graphics taking on a different scale of meaning. The photo spread on the reverse of the poster demonstrates this effect.



Starting the Program

When the program is RUN, you will be asked if you desire to load in a new map. A positive response will cause the program to ask you for the map's filename to be loaded into memory. Otherwise, a blank map is created. In either case, control then passes to the main menu.

Main Menu

From this menu, all the different parts of the program are reached. The menu gives six options: "Draw" goes to the "Map Creator;" "Map Save/Load" goes to the disk I/O routine; "Graphic Draw" allows a parcel to be defined; "Print Map" is used to print all or part of the map; "Jump" allows a jump to be added to the jump list; and "Print Graphics" prints the parcels onto a printer.

Map Creator

The "Map Creator" display shows a portion of the map, the current cursor character in the upper right, and a menu of commands near the bottom of the screen. These commands are activated by simultaneously depressing the "Shift" and "Down-arrow" keys (together, they act as a control key) as well as the character indicated in the menu. There is also a flashing cursor showing your current position, as well as legends along the x and y axes to show the position of each square within the large 99 by 99 map.

The cursor is moved using the four arrow keys. If you move off the boundaries of the screen, the portion of the map on the screen will change to reflect your new position. To place a character on the map, simply type the character. If the character has already been defined as a tile, it will then be put on the map. Otherwise, you will be allowed to create it before it is placed on the map. Another way to place a character is to hit the "Enter" key, which will repeat the last character placed on the map.

The list of commands at the bottom of the screen is executed by typing "Shift/Down-Arrow" and the letter. This is called "Control G" (or whatever letter).

Control G — Allows you to enter a character on the map that cannot be typed from the keyboard, such as an arrow (see characters below).

Control P — Allows you to reposi-

tion the cursor by specifying the x and y coordinate of where you wish to go.

Control S — Returns to the main menu.

Control O — Produces a hardcopy printout of what is on the screen.

Control D — Allows you to examine parcels. After typing "Control D", you will be asked for the character to be displayed. Upon typing it, the parcel for that character will be displayed on the right of the screen, and you will be asked for another character. Hitting "Enter" will escape from this mode.

Control C — Changes the graphic mode that is displayed on the screen. In modes above 0, the graphics for each character are displayed instead of the characters themselves. Otherwise, everything is the same. See poster for a description of the graphic modes.

Clear — Toggles the automatic repeat on or off. When it is on, the cursor is moved in the direction of the last arrow when a character is placed on the map.

Map Save/Load

This routine displays a short menu. The four options allow you to load a map, save a map, delete a file from disk, or return to the main menu. Loading a map will erase any map currently in memory, so be careful. If the disk does not have room to save the map, the delete file can be used to make room. If you make a mistake, such as trying to load a nonexistent file, an error will result. Type "GOTO 150" to return to the main menu.

Draw

The "Graphic Draw" routine allows you to define the graphic tile for a character. It is similar in some ways to "Map Creator" above, with a few differences. The size of the tile set is much smaller (9 x 6). Therefore, the entire parcel can be displayed on the screen at one time. There is also a diagram showing the codes for the six pixels in a character to aid in computing graphic values. Cursor movement is the same as "Map Maker," and is accomplished with the typing of characters.

The special commands for "Graphic Draw" are:

Control F — Fills the parcel with a specified character.

Control G and Control S — The same as in "Map Maker."

Control E — Loads a graphic tile from another character into the current parcel (useful for creating numerous similar parcels which are slightly different from each other).

Controls U, D, R and L — Draw lines of a specified character up, down, right or left from the current cursor position.

Clear — Toggles the repeat on/off, as in "Map Maker."

Control T — Changes the function. This allows you to change the cursor to a graphic pixel, and be able to draw, erase or move in single or double pixel groups. When used in conjunction with the repeat, excellent graphic designs can be created with relative ease.

Print Map

First, you will be asked if you want condensed print and what type of printer you have. The required print control code will then be printed. If you have an Epson MX-80, you must have it set to the S-80 mode for the print routine to function. If you have neither an MX-80 nor a Microline 83, substitute your printer's codes for condensed and normal print for the pairs of X's in line 8003.

Next, you will be asked which graphic mode is to be printed and the coordinates of the block of map to be printed. Enter the coordinates of the upper left and lower right points in "x,y" format.

Finally, you will be asked the width of your printer. This is used as a check to make sure that what you requested to print will fit on your printer.

Jumps

This routine allows you to enter a jump into the jump table. In this program, it has no real purpose. The jumps are only used by GAMEPLAY/BAS. A jump is entered in the form "xyyxyy" with the first four characters as one coordinate and the last four as the second coordinate. A jump will take you from the first coordinate to the second, or vice versa.

Print Graphics

After asking you about condensed printing, as in the "Print Map" routine, this routine will ask you for the first and last characters, and will then print all tiles within that range.

continued on next page

continued from previous page

```
110 CLEAR24000:DIMMP$(154)
112 DIMXP(3),YP(3),XS(3),YS(3):XS(0)=1:YS(0)=1
113 GOSUB20000
115 KB=14400:MD=0
150 Q=0:DI=2:C1=0:C2=0:RL=1:RC=1:PL=1:PC=1:MD=0:CH#="" :CU#="Q":C
LS:PRINT@20,"MAIN MENU
Draw 1
Map Save/Load 2
Graphic Draw 3
Print Map 4
Jump 5
Print Graphics 6
":INPUT"Selection >";Q
151 D2=2:RP=-1:X9=1:Y9=1:69=0:M9=3:ONQGOSUB290,500,40000,9000,31
000,9500
152 GOTO150
199 CLS:PRINT"Saving NEWMAP/BAS:0";SAVE"NEWMAP/BAS:0":PRINT"Savi
ng NEWMAP/BAS:1";SAVE"NEWMAP/BAS:1":STOP
200 '
210 CLS:IFMD=0THENFORI=RLTORL+10:PRINTMID$(MP$(I),RC,50);I:NEXTI
:ELSEGOSUB5000:GOTO230
220 PRINTTAB(0);STR$(RC);TAB(8);RC+10;TAB(18);RC+20;TAB(28);RC+3
0;TAB(38);RC+40;TAB(48);RC+50
230 PRINT@768,"Arrows move cursor <Enter> Repeats characte
r
G-Graphic char P-Position cursor S-Save and go to Menu";
240 PRINT@896,"D-Hardcopy D-Display char C-Change
Mode (";MD;");CHR$(30)
244 PRINT@960,"Clear-Repeat ";MID$("Dff---Dn ",RP*3+4,3);
250 RETURN
260 CLS:FORI=RLTORL+10:PRINTMID$(HL$(I),RC,50);I:NEXTI:GOTO220
290 GOSUB200
298 X=PEEK(15360+C2*64+C1):IFM9=3THENIFCU#=#CHR$(X) THENX1=ASC(CU#
)ELSEX1=32ELSEX1=128+2[(Y9*2+X9-3)
299 PRINT@54,"Cursor=";MID$(STR$(X),2);" ";:IFPEEK(14400)AND2THE
NIFC9#<>CHR$(31) THENCH#=#C9#:GOTO301
300 PRINT@C2*64+C1,CHR$(X1);:FORI=1TO10:CH#=INKEY#:IFCH#<>""OR(P
EEK(KB)<>0ANDPEEK(KB)<>16) THEN301ELSENEXTI:PRINT@C2*64+C1,CHR$(X
);:FORI=1TO10:CH#=INKEY#:IFCH#<>""OR(PEEK(KB)<>0AND(PEEK(KB)<>16
ORPEEK(14464)=0) THEN301ELSENEXTI:GOTO299
301 PRINT@C2*64+C1,CHR$(X);:C9#=#CH#:IFM9=3THEN307
302 IF69<>0THEND=2[(Y9*2+X9-3):IFM9=2THEND=0+2[(Y9*2-1)
303 IF69<>0THENX=-X*(X>127)-128*(X<128):IF69=1THENCH#=#CHR$(XANDD
):ELSECH#=#CHR$(XORD)
304 IF69<>0THENPRINT@C2*64+C1,CH#;:MID$(MP$(PL),PC,1)=CH#;CU#=#CH
#;X=ASC(CH#)
307 EX=0:IFCH#<>""THENEX=ASC(CH#)
308 XS=XS(MD);YS=YS(MD)
310 IFEX=9OREX=BOREX=10OREX=91ORPEEK(KB)AND8ORPEEK(KB)AND32ORPEE
K(KB)AND64OR(PEEK(KB)AND16ANDPEEK(14464)=0) THENPRINT@C2*64+C1,CH
R$(X);:GOSUB400:GOTO298
315 IFEX=13THENCH#=#CU#
318 IFCH#=#CHR$(19) THENRETURN
320 IFCH#=#CHR$(16) THENPRINT@760,CHR$(31);:PRINT@832,"Enter X pos
ition desired (1-");MID$(STR$(M9*9),2);" ";:INPUTCH:IFCH#>M9*9ORCH
<1 THENGOTO320ELSEPC=CH
325 IFCH#=#CHR$(16) THENPRINT@832,"Enter the Y position desired (1
-");MID$(STR$(M9*9),2);" ";:INPUTCH:IFCH#>M9*9ORCH<1 THENGOTO325ELSE
EPL=CH
326 IFCH#=#CHR$(16) THEN344
330 IFCH#=#CHR$(7) THENPRINT@896,CHR$(31);"Enter character >";:GOS
UB41200:CH#=#Q#:GOSUB240:GOTO307
340 IFCH#<>CHR$(3) THEN350
342 PRINT@896,CHR$(31);"Enter mode (0-3) ";:LINEINPUTQ#:MD=VAL(Q
#)
343 IFMD<>0ANDXP(MD)=0THEN342
344 RC=PC-INT(27/XS(MD)+.51):IFRC<1THENRC=1
345 RL=PL-(6/YS(MD)):IFRL<1THENRL=1
```

```
346 C2=(PL-RL)*YS(MD):C1=(PC-RC)*XS(MD):IFMD=0THEN290ELSEGOSUB23
0:FORI=0TO11:PRINT@I*64,STRING$(54," ");:NEXTI:GOSUB5000:GOTO298
350 IFCH#=#CHR$(15) THENA#="" :Q=VARPTR(A#):POKEQ,54:FORI=0TO11:POK
EQ+2,60+I/4:POKEQ+1,(I-INT(I/4))*64:LPRINTA#:NEXTI:GOTO299
360 IFCH#=#CHR$(4) THENPRINT@896,CHR$(31);"Enter character (<Enter
> to quit) >";:GOSUB41200:IFQ1#="" THENGOSUB240:GOTO298ELSECH#=#Q#
:GOSUB2000:X=247:PRINT@X-64,"Char=";MID$(STR$(ASC(Q#)),2);" ";:G
OSUB41050:CH#=#CHR$(4):GOTO360
375 IFCH#=#CHR$(31) THENRP=-RP:GOSUB244:GOTO299
387 IFCH#<" " THEN298
390 MID$(MP$(PL),PC,1)=CH#;CU#=#CH#:IFMD=0THENPRINT@C2*64+C1,CH#;
:X=ASC(CH#)
391 IFCH#>=#CHR$(128) THEN299ELSEGOSUB2000:IFMID$(MP$(R1+TY-1),R2+
TX-1,1)="" THENPRINT@896,CHR$(31);"Character not defined. Do it
now";:INPUTA#:IFLEFT$(A#,1)="" THENCLS:GOSUB40001:GOTO290:ELSEG
OSUB240:GOTO299
392 IFMD=0THEN299ELSEFORI=1TOYS:IFC2+I-1=>11 THEN298
393 IFC1+XS>49 THENPRINT@ (C2+I-1)*64+C1,MID$(MP$(R1+YP(MD)+I-2),R
2+XP(MD)-1,50-C1);:GOTO396
394 PRINT@ (C2+I-1)*64+C1,MID$(MP$(R1+YP(MD)+I-2),R2+XP(MD)-1,XS)
;
396 NEXTI
398 IFRP=-1 THEN298ELSECH#=#MID$(CHR$(8)+CHR$(9)+CHR$(10)+CHR$(91)
,DI,1):X=PEEK(15360+C2*64+C1):GOTO307
400 IF(EX=8ORPEEK(KB)AND32)ANDPC-1>=1 THENX9=X9-INT(M9/2+1):IFX9<
1 THENX9=1-(M9=1):DI=1:PC=PC-1:C1=C1-XS:IFC1<0 THENRC=RC-27/XS:C1=
C1+27:IFRC<1 THENC1=C1+(1-RC)*SX:RC=1:GOSUB200ELSEGOSUB200
410 IF(EX=9ORPEEK(KB)AND64)ANDPC+1<=M9*9 THENX9=X9+INT(M9/2+1):IF
X9>2 THENX9=1:DI=2:PC=PC+1:C1=C1+XS:IFC1>49 THENRC=RC+27/XS:C1=C1-
27:IFRC>M9*9 THENC1=C1-(RC-M9*9)*SX:RC=M9:GOSUB200ELSEGOSUB200
420 IF(EX=10OR(PEEK(KB)AND16)ANDPEEK(14464)=0) ANDPL+1<=M9*9 THE
NY9=Y9+1-2*(M9=3):IFY9>3 THENY9=1:DI=3:PL=PL+1:C2=C2+YS:IFC2>10 TH
ENRL=RL+6/YS:C2=C2-6:GOSUB200
430 IF(EX=91ORPEEK(KB)AND8)ANDPL-1=>1 THENY9=Y9-1+2*(M9=3):IFY9<1
 THENY9=3+2*(M9=3):DI=4:PL=PL-1:C2=C2-YS:IFC2<0 THENRL=RL-6/YS:C2=
C2+6:GOSUB200
440 RETURN
500 CLS:PRINT@20,"DISK READ / SAVE
```

```
Load Map 1
Save Map 2
Delete File 3
Return to Menu 4
":INPUT"Selection";Q:ONQGOSUB505,520,560:RETURN
```

```
505 LINEINPUT"Name of file to be loaded >";Q#:CLOSE:OPEN"1",Q#
508 INPUT#1,BC,FM,JR,LC,MH,MW,NE,NG,NR,PV,SG,SX,SY,TX,TY,GC,N1,N
2:GOSUB6000:I=1
510 IFEOF(1) THENCLOSE1:RETURNELSELINEINPUT#1,MP$(I):I=I+1:GOTO51
0
520 LINEINPUT"Name of file to be saved >";Q#:CLOSE:OPEN"0",Q#
523 PRINT#1,BC,"FM","JR","LC","MH","MW","NE","NG","NR","PV","SG
","SX","SY","TX","TY","GC","N1","N2
525 FORI=1TO154:PRINT#1,MP$(I):NEXTI:CLOSE1:RETURN
560 LINEINPUT"Name of file to be deleted >";Q#:KILLQ#:RETURN
2000 E=ASC(CH#)-BC:R1=((INT(E/MH)*SY)+100:R2=SX*(E-INT(E/MW)*M
W))+1:IF(R1<100)OR(R1>130) THENR1=100:R2=1:RETURNELSERETURN
4009 STOP
5000 I=RL:I1=0
5010 Q=RC:Q1=0
5020 REH
5030 CH#=#MID$(MP$(I),Q,1):GOSUB2000
5040 PRINT@Q1*64+I1,;:
5042 IFQ1+T>11 THEN5070
5045 Z2=R1+YP(MD)-1:Z3=R2+XP(MD)-1:IFI1+XS(MD)>50 THENZ1=50-11ELS
EZ1=XS(MD)
5046 A#=#CHR$(26)+STRING$(Z1,24)
```

```

5047 IFMD=1THENPRINTMID$(MP$(Z2),Z3,Z1)A$;MID$(MP$(Z2+1),Z3,Z1)A
$;MID$(MP$(Z2+2),Z3,Z1);ELSEIFMD=2THENPRINTMID$(MP$(Z2),Z3,Z1)A
$;MID$(MP$(Z2+1),Z3,Z1);ELSEIFMD=3THENPRINTMID$(MP$(Z2),Z3,Z1);
5070 IFI1+XS(MD)>50THENI1=0;Q1=Q1+YS(MD);IFQ1<11THENI1=1+I;Q=RC:G
0T05020ELSE5100
5080 I1=I1+XS(MD);Q=Q+1;GOTO5030
5100 Q=RL-1;FORI=0T010:PRINT$(I)*64+51,;:IFI/YS(MD)=INT(I/YS(MD)
)THENQ=Q+1;PRINTQ;ELSEPRINT " ";
5110 NEXTI
5115 PRINT@704,CHR$(30);
5120 Q=RC-1;FORI=0T049STEPXS(MD):PRINT@704+I,USING"###";Q+1;:Q=Q+
1:NEXTI
5190 RETURN
6000 SX=9;SY=6;TX=7;TY=5
6010 XP(1)=1;YP(1)=1;XS(1)=9;YS(1)=3;XP(2)=4;YS(2)=6;YS(
2)=2;XP(3)=7;YP(3)=4;XS(3)=3;YS(3)=1
6140 RETURN
8000 CLS
8001 INPUT"Do you want condensed mode (Y/N)";Q$
8002 INPUT"
Enter printer type:

Microline 80 or 83 1
Epson MX-80 2
Other 3

Your choice >";Q1
8003 IFLEFT$(Q$,1)="Y"THENLPRINTMID$(CHR$(29)+" "+CHR$(29)+" XX"
,Q1*2-1,1);ELSELPRINTMID$(CHR$(30)+" "+CHR$(30)+" XX",Q1*2-1,1);
8004 RETURN
9000 GOSUBB000
9005 PRINT"
Enter graphics mode (0-3) >";:INPUTPM
9010 XS=XS(PM);YS=YS(PM):PRINT"Enter X1,Y1,X2,Y2 >";
9020 XP=XP(PM);YP=YP(PM):INPUTZ1,Z2,Z3,Z4
9030 INPUT"Enter printer width >";PW
9032 IF(Z3-Z1+1)*XS>PW-4THENPRINT"Won't fit!";:GOTO9005
9040 FORI=1T03:LPRINT " ";:
9050 FORQ=Z1T0Z3:Q$=MID$(STR$(Q),2):Q$=STRING$(3-LEN(Q$)," ")&Q$
:LPRINTMID$(Q$,1,1);STRING$(XS-1," ");
9060 NEXTQ:LPRINT " ";NEXTI
9070 FORQ=Z2T0Z4:LPRINTUSING"### " ;Q;
9080 IFPM=0THENLPRINTMID$(MP$(Q),Z1,Z3-Z1+1):NEXTQ:RETURN
9085 FORQ1=1T0YS;IFQ1<>1THENLPRINT " ";
9090 FORI=Z1T0Z3:CH$=MID$(MP$(Q),I,1):GOSUB2000
9100 LPRINTMID$(MP$(R1+Q1+YP-2),R2+XP-1,XS);
9110 NEXTI:LPRINT " ";NEXTQ1,Q
9120 RETURN
9500 GOSUBB000:CLS:PRINT"Starting Character >";:GOSUB41200:Z1=AS
C(Q$):PRINT"Ending Character >";:GOSUB41200:Z2=ASC(Q$)
9510 FORT=Z1T0Z2
9520 CH$=CHR$(T)
9530 GOSUB2000:R2=R2-1
9540 LPRINTMID$(MP$(R1),R2+1,9);" ";MID$(MP$(R1+3),R2+1,6);" "
;MID$(MP$(R1+3),R2+7,3)
9550 LPRINTMID$(MP$(R1+1),R2+1,9);" ";MID$(MP$(R1+4),R2+1,6);" "
;MID$(MP$(R1+4),R2+7,3)
9560 LPRINTMID$(MP$(R1+2),R2+1,9);" ";MID$(MP$(R1+5),R2+1,9)
9570 LPRINT " "
9580 NEXTT
9590 RETURN
9999 STOP
10000 CLS
10002 PRINT"Map= 1,1 to ";MID$(STR$(MH*9),2);";";MID$(STR$(MH*9)
,2)
10022 FORI=1T03:LPRINT " ";FORQ=Z1T0Z3:Q$=MID$(STR$(Q),2)
10024 Q$=STRING$(3-LEN(Q$)," ")&Q$:LPRINTMID$(Q$,I,1);
10026 NEXTQ:LPRINT " ";NEXTI

```

```

10030 FORQ=Z2T0Z4
10040 LPRINTUSING"### " ;Q;:LPRINTMID$(MP$(Q),Z1,Z3-Z1+1)
10050 NEXTQ
10060 RETURN
20000 CLS
20005 BC=32;MW=11;MH=11;NR=3;JR=1;NE=1;EW=29;FM=1;NC=1;GC=1
20012 INPUT"Do you wish to edit an old map";A$;IFLEFT$(A$,1)="Y"
THENB0T0505
20013 FORI=1T0154:MP$(I)=STRING$(99,".");NEXTI
20158 GOSUB6000
20999 RETURN
31000 CLS
31010 PRINT" JUMP DEFINITION
"
31020 PRINT"Enter jump number (Last=";MID$(STR$(N2),2);", Max=";
MID$(STR$(JR*MW),2);")";
31030 INPUTC:IFC>N2THENN2=C
31040 S1=SX;S2=SY;R1=NR*SY+MH*9+INT((C-1)/MW)+1;R2=((C-1)-INT((C
-1)/MW)*MW)*9+1;SX=9;SY=1;PRINT@64,CHR$(31);:GOSUB40002:SX=S1;SY
=S2;RETURN
40000 CLS:LINEINPUT"Which character do you wish to prepare >";CH
$
40001 GOSUB2000
40002 H$="0";PRINT"Line Ref";R1,"Char Ref";R2;GOSUB41100
40003 CX=202;XL=R1;XC=R2;X=138;GOSUB41050
40004 H1$=MID$(MP$(XL),XC,1);IFM9<>3THENH$=CHR$(128+2*(Y9*2+X9-3
))
40005 PRINT@CX,H$;:CU$=INKEY$;IF(CU$="")AND(PEEK(KB)=00R(PEEK(KB
)=16ANDPEEK(14464)<>0))THENPRINT@CX,H1$;:GOTO40005
40009 IFCU$=CHR$(13)THENCU$=H$
40010 IF(CU$=CHR$(8)ORPEEK(KB)AND32)AND(XC-R2<ORX9-INT(M9/2+1)>
0)THENGOSUB41300:D2=1;GOSUB41040;X9=X9-INT(M9/2+1);IFX9<1THENX9=
1-(M9=1);XC=XC-1;CX=CX-1;GOTO40004
40020 IF(CU$=CHR$(9)ORPEEK(KB)AND64)AND(XC-R2<SX-1ORX9+INT(M9/2+
1)<=2)THENGOSUB41300:D2=2;GOSUB41040;X9=X9+INT(M9/2+1);IFX9>2THE
NX9=1;XC=XC+1;CX=CX+1;GOTO40004
40030 IF(CU$=CHR$(10)OR(PEEK(KB)AND16ANDPEEK(14464)=0))AND(XL-R1
<SY-1ORY9+1-2*(M9=3)<=3)THENGOSUB41300:D2=3;GOSUB41040;Y9=Y9+1-2
*(M9=3);IFY9>3THENY9=1;XL=XL+1;CX=CX+64;GOTO40004
40040 IF(CU$=CHR$(91)ORPEEK(KB)AND8)AND(XL-R1>0ORY9-1+2*(M9=3)>0
)THENGOSUB41300:D2=4;GOSUB41040;Y9=Y9-1+2*(M9=3);IFY9<1THENY9=3+
2*(M9=3);XL=XL-1;CX=CX-64;GOTO40004
40042 IFCU$=" "ANDCU$<>"I"THENGOSUB41030:IFRP=1THENCU$=MID$(CHR
$(8)+CHR$(9)+CHR$(10)+CHR$(91),D2,1);GOTO40009ELSE40004
40050 IFCU$=CHR$(19)THENRETURN
40060 IFCU$=CHR$(7)THENPRINT@B00,"Enter character >";:GOSUB41200
:CU$=Q$;GOTO40042
40065 IFCU$<>CHR$(20)THEN40070
40066 PRINT@B96,CHR$(31);"<S>ingle pixel, <D>ouble pixel, <N>orm
al type >";:LINEINPUTA$;A=INSTR("SDN",LEFT$(A$,1));IFA=0THEN400
66ELSEM9=A;IFA=NTHEN40068
40067 PRINT@B96,CHR$(31);"<S>et, <R>eset, <M>ove >";:LINEINPUTA
$;A=INSTR("MRS",LEFT$(A$,1));IFA=0THEN40067ELSEG9=A
40068 PRINT@B32,CHR$(31);:GOSUB41101
40070 IFCU$=CHR$(6)THENPRINT@B00,"Filler character >";:GOSUB4120
0:CU$=Q$;FORQ=0T0SY-1:MID$(MP$(R1+Q),R2,SX)=STRING$(SX,CU$);NEXT
Q;GOTO40003
40072 IFCU$=CHR$(16)THENH$=MID$(MP$(XL),XC,1);GOTO40004
40075 IFCU$=CHR$(5)THENZ1=R1;Z2=R2;PRINT@B00,"Enter character >
";:GOSUB41200:CH$=Q$;GOSUB2000:FORI=0T0SY-1:MID$(MP$(Z1+I),Z2,SX)
=MID$(MP$(R1+I),R2,SX);NEXTI;R1=Z1;R2=Z2;GOTO40003
40077 IFCU$=CHR$(31)THENRP=-RP;GOSUB41101;GOTO40004
40080 A=INSTR(CHR$(21)+CHR$(4)+CHR$(12)+CHR$(18),CU$);IFA=0ORCU$
=" "THEN40200
40082 PRINT@B96,"Enter draw character >";:GOSUB41200:CU$=Q$
40083 PRINT@CX,Q$;MID$(MP$(XL),XC,1)=Q$
40085 IFA=1ANDXL-R1>0THENXL=XL-1;CX=CX-64;GOTO40083

```

continued on next page

continued from previous page

```
40090 IFA=2ANDXL-R1<SY-1THENXL=XL+1:CX=CX+64:GOTO40083
40094 IFA=3ANDXC-R2>0THENXC=XC-1:CX=CX-1:GOTO40083
40098 IFA=4ANDXC-R2<SX-1THENXC=XC+1:CX=CX+1:GOTO40083
40200 GOTO40004
41030 PRINT@CX,CU#;:MID$(MP$(XL),XC,1)=CU#;H#;CU#;RETURN
41040 PRINT@CX,MID$(MP$(XL),XC,1);:RETURN
41050 PRINT@X,STRING$(SX,138);:X=X+63:FORI=0TOSY-1:PRINT@X,CHR$(
133);MID$(MP$(R1+1),R2,SX);:X=X+64:NEXTI:RETURN
41100 PRINT@576,"F-Fill      G-Graphic char  S-Save & Menu
E-Enter Graphic";
41101 PRINT@640,"U-Draw up   D-Draw down    R-Draw Right
L-Draw left
Clear-Repeat (";MID$("Off)----On)",RP#4+5,4);
41102 PRINTTAB(30);"T-Function (";MID$("SingleDoubleNormal",M9#6
-5,6);:IFM9=3THENPRINT");ELSEPRINT " ;MID$("Move) Reset)Set) "
,69#6-5,6);
41103 PRINT"
```

```
P-Pick up character";
41105 PRINT@290," 1 2";:PRINT@354," 4 8";:PRINT@418,"16 32";
41110 RETURN
41200 X=PEEK(16416)+PEEK(16417)*256-15360:PRINTCHR$(30);:LINEINP
UT@#:Q1#;Q#
41210 ILEFT$(Q#,1)<>"?THENIFLEN(Q#)>1THENO=VAL(Q#):GOTO41260EL
SEIFQ#<>"?THENO=ASC(Q#):GOTO41260
41220 Q=128:Q#;MID$(Q#,2)
41230 IFQ#=""THEN41260
41240 Q1=VAL(Q#):Q#;MID$(Q#,LEN(STR$(Q1)))
41250 Q=Q+Q1:GOTO41230
41260 IFQ<32ORQ>191THENPRINT@X,;:GOTO41200
41265 Q#;CHR$(Q):RETURN
41300 IFM9=3OR69=0THENRETURNELSEO=2I(Y9#2+X9-3):IFM9=2THENO=O+2I
(Y9#2-1)
41310 X=ASC(MID$(MP$(XL),XC,1));X=-X*(X>127)-128*(X<128):IF69=3T
HENX=XORDELSEX=XAND(255-O)
41320 MID$(MP$(XL),XC,1)=CHR$(X):RETURN
```



Envyrn™

Its Marketing

A statement from its author and publisher of
SoftSide, Roger W. Robitaille, Sr.

It would be possible to complete **Envyrn's™** features as they now exist and offer **Envyrn-I** or its equivalent. However, I've seen too much significant software get caught in the revision bind. It's not uncommon for a software publisher to offer what appears to be a completed product and find himself in an endless series of updates and product upgrades with trade-in allowances. I'd rather be more straightforward concerning the process of software development and offer the public a participation in the project. The vital difference is that we make no pretense that a finished product exists at this time, or that one may exist for several years — if ever. Rather, we offer that prototypes are constantly being refined and will be sent to participants on at least a quarterly basis, including any design notes, changes in documentation, etc. If participation is heavy, we will be able to take a more organized approach to keep participants advised through better documentation. Participation will cost \$200 per year. Participants are invited to offer enhancements to the features of **Envyrn™** by way of suggestions and code. All mail will be read and most answered. We expect some employment opportunities to open due to **Envyrn's™** development. Please contact our offices should you be interested.

In order to clearly understand the meaning of **Envyrnments™**, you must appreciate that **Envyrn™** databases may be very significant by themselves. Database managers, word processors, and Visicalc™ tend to be utilized for

private purposes. When a word processor is used to develop a book, it is seldom distributed on diskette. Even a small book would involve several diskettes and cost \$50 or more. A system of form letters would comprise a database significant in itself or as a database for word processors. **Envyrn™** has the capability of preparing, editing, printing and storing the likes of electrical circuit designs, road maps, and floor plans. The diagrams it stores may be projected on large-screen TV, preserving the very blackboard of today's education. They can be edited or updated at any time. **Envyrn™** is a program all school systems should have. **Envyrn™** invites the first generation of computer novelists to begin honing their skills. As many of the databases will be important in and of themselves, they deserve a term uniformly suggesting the approach they use. To secure quality standards, the term **Envyrnment™** is reserved as a mark of quality we control.

The impact of **Envyrn™** will extend far beyond our walls. Our goal is to cohesively develop **Envyrn™** in its larger context, as uniformly as possible. **Envyrn™** databases may be offered by anyone, but **Envyrnments™** have been reviewed by **SoftSide Publications** and found to be compatible and useful.

It is necessary to prepare a library approach for **Envyrnments™**. There is every reason to assume that **Envyrn's™** approach will take three to five years to develop. As with discreet programming itself, only so many **Envyrnments™**

will have a substantial commercial future. Most will have to be satisfied with availability only through the library. It's substantially significant that these efforts be readily available. As part of our responsibility, we'll develop as efficient and available a library as possible.

The library policies (subject to change as experience dictates) will be as follows:

1. A database must pass review as an **Envyrnment™**. If the database joins the **Envyrnment™** library, there will be no charge. Should the author choose not to have the database enter the library, there will be a \$100 certification fee. For the moment, there will be no charges for rejections. Should authors wish to market their own **Envyrnments™**, we wish them the best and they're free to use the term **Envyrnment™** with regard to databases certified by **SoftSide Publications**. Our acceptance must bear the year of certification — **Envyrnment™-1981** and updates must be recertified (reduced fee).

2. With the author's approval, his database will enter the library and a thorough description will be prepared for the library directory. Authors are encouraged to add a user's fee to be charged for their work when ordered by a subscriber. Anyone charging a user's fee over \$100 should be prepared to give telephone assistance to anyone who fairly procured a copy of his **Envyrnment™**. Public domain donations are permissible.

3. Documentation (user instructions)

must be prepared in machine readable form, on disk. We must shield the library from administrative nightmares. One of the policies will be not to store printed instructions. This will make updating a far simpler process.

4. Any registered participant of **Envyrn™** may utilize the library. If you are a current participant, you will receive a subscription to the **Envyrnment™ Library Directory** as part of your membership. Subscriptions to the directory will cost \$5 per year (2-4 issues).

5. The **Envyrnment™ Resource Library** will charge \$10 per diskette required plus a usage fee(s) for its service.

6. Popular library items will (with the author's approval) be slated for commercial development.

7. Databases may be withdrawn from the library with a six month written notice (to make sure it's deleted from the current directory) and payment of a review fee.

Diversions™ thru Envyrn™ will also use the library as one of its resources. **Diversions™** will suggest the impact realm of **Envyrn™**. A special companion program is prepared to respond to **Envyrnments™** on a wide spectrum of applications. Some are essentially

for amusement, like graphic adventures. Some may dynamically illustrate more serious concepts, such as traffic patterns and mapping techniques. Other related material, including techniques and graphic hints, will be a part of **Diversions™**. Mostly, **Diversions™** will be about the **Envyrnments™**. If the **Envyrnment's™** about oil drilling and robbing a bank, the articles will refer to oil drilling and bank robbery.


For the moment, we promise one database with a supporting interpreter every two months. The six issues per year will each consist of one disk, containing the appropriate software, and sixteen or more pages devoted to whatever supporting textual and pictorial information is needed to complete the offering. The ultimate plan involves two or three **Envyrnments™** per issue. Initially, the subject matter will be selected primarily for entertainment value. Subjects will include "Star Trek," role-playing mysteries, and the like. Later, educational subjects will be introduced to enhance the entertainment theme. We hope to have a periodical to rival **SoftSide** qualitatively by 1983. The cost of a subscription to **Diversions™** is \$10 per issue — \$60 per year. Back issues will be available for \$20 per copy. We are certain you

will be pleased with the value you receive.

There you have it. You can:

A. Join the **Envyrn™** software participation group. You'll receive quarterly updates of the **Envyrn™** editor and project notes. You'll also get the **Envyrnment™ Library Directory** when it is available. The cost is \$200 per year. Your first installment will be mailed in January of 1982.

B. Subscribe to **Diversions™** thru **Envyrn™** for only \$60 per year. We guarantee an experience you've heretofore only dreamed of. The premier issue for the TRS-80 Models I and III is projected to be available in December of 1981. An Apple version is expected to be released in March of 1982.

Envyrn™ is a copyrighted program with all rights reserved by its author, **Roger W. Robitaille Sr.** **Envyrn™** is an extremely significant breakthrough in computer software and will be treated as such by its owner. The **Envyrn™** editor prototype and the **GAMEPLAY** interpreter, included in this issue and on TRS-80 DV, are for private use only. Any other use of these programs, for whatever purpose, is subject to specific written approval from the author and **SoftSide Publications.** 

Envyrn™

GAMEPLAY/BAS

Instructions for S-80 DV

GAMEPLAY is the prototype of the interpreter being prepared for **Diversions™**, the upcoming **Envyrn™**-based magazine. It controls the execution of specially prepared **Envyrnments™** and the player's movement through them.

When the program is run, you will be presented a small menu with three options. The first allows you to enter the **Envyrnment™** currently in memory. The second allows you to save an **Envyrnment™** in progress, and the third loads a new **Envyrnment™** from diskette. In most instances, you'll first want to load the database on disk called **ADV/DAT**, then enter it.

After doing this, you will see a display showing where you are, as well as displays giving your facing, activity level and energy. Usually, you'll be shown only the tile you occupy (flashing) and the tiles immediately

surrounding your position. Once you've viewed a tile, however, you'll be able to see it from any distance, as long as it will fit within the screen parameters. Those with little patience may use the **CHEAT** command (see below). The function controlling the exposure of tiles will be disabled, and the screen will fill completely with tiles.

These are the operating commands presently active in the program. Their functions are:

@ = Enter as prefix to long command (see below)

(<) = Decrease body activity (slower)

(>) = Increase body activity (faster)

S = Change body activity to swim
Left Arrow = Turn facing counterclockwise

Right Arrow = Turn facing clockwise

- = Return to menu
U = Use something (such as a door-open/close)
E = Enter (a building)
I = Identify the parcel directly in front of you

Long commands

Hitting the @ symbol puts you in a mode in which you can type multiple letter commands. Only the first three letters are significant, although the entire command may be typed, if desired.

LOOK(K): Change display to graphic mode one

SEE: Graphic mode two


WAT(CH): Graphic mode three

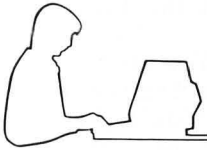
IDE(NTIFY): Same as I above

ENT(ER) or GO: Same as E above

SWI(M): Same as S above

CHE(AT): Disable limited display (see above)

NOR(MAL): Disable **CHEAT** function 



SUPER DAIRY FARMING

Apple version by Jean H. Anderson
original program by Scott Tapley

"Super Dairy Farming" is a simulation for a 48K Apple with Applesoft and disk drive. It is included as a bonus program on this month's Apple Disk Version of SoftSide.

This simulation first appeared as an S-80 program in the May issue of **SoftSide**; an Atari translation was published in August. Now Apple disk subscribers, too, can exercise their agri-business skills with this enhanced translation.

Beginning with half a million dollars, your goal is to accumulate 10,000 points, based on your total assets of land, cows, money, etc. You may choose to play with or without the "natural disaster" option, which subjects you to such goodies as disease, tornadoes, lightning, hailstorms, and drought. After buying a farm, you proceed to the three-phase daily cycle of the simulation.

In Phase One you care for the cows

in various ways: feeding, milking, buying, and selling. In this version the cows need to be milked twice each day, or milk production will suffer. Feeding them is also a priority, obviously: You can do that by letting them graze or by feeding them silage. Grazing costs nothing, but it does deplete the grass, and the pasture must be allowed to recover occasionally. If you fail to feed the cows, one will die later in the day, and they will produce inferior milk: When you have accumulated enough milk (at least 200 gallons), you can sell some or all of it to the milkman.

In Phase Two you can buy and sell capital goods. You can buy or sell land; buy a cooler, a barn, or a silo; or even sell the whole farm if you want out. Except for selling land, all these things help to accumulate points.

In Phase Three the computer calculates your points and financial status, and if you have an outstanding loan you may make a payment on it.

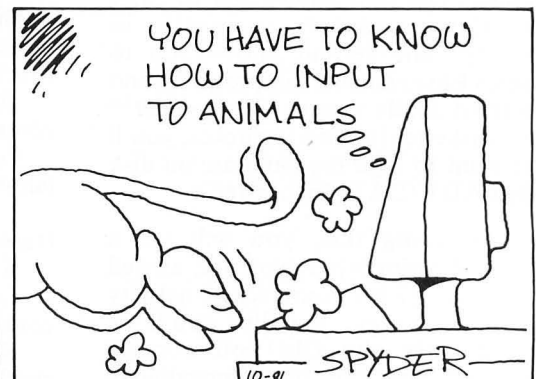
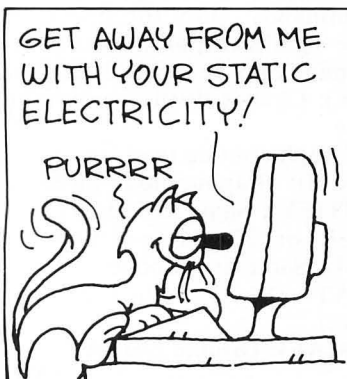
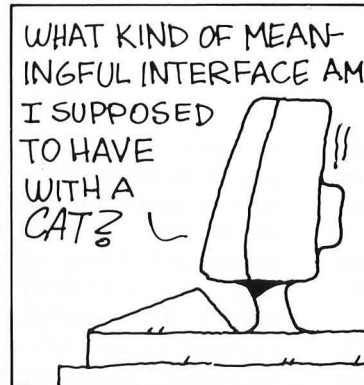
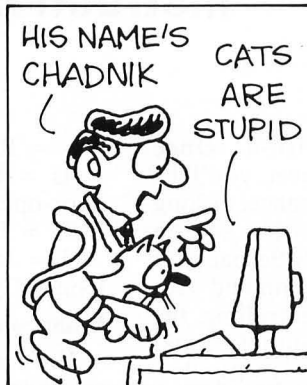
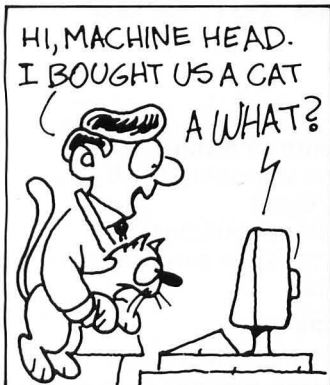
Your points are calculated as follows:

- One point for every:
 - \$10,000 in cash;
 - 5 acres of land;
 - 500 gallons of milk;
 - Small cooler;
 - 7 Jersey cows;
 - 6 Guernsey cows;
 - 5 Ayrshire cows;
 - 4 Brown Swiss cows;
 - 3 Holstein cows.
- Two points for every large cooler.
- Four points for every small silo.
- Five points for every barn or medium silo.
- Six points for every large silo.
- Minus one point for every \$100 borrowed.

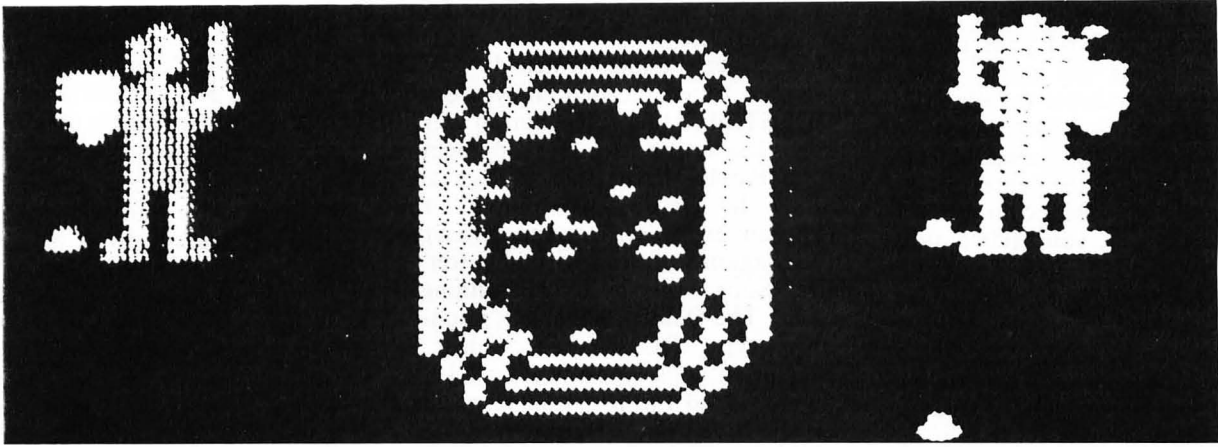
Lacking the necessary 10,000 points, you then begin a new day with the same three-phase format. When you do reach your goal (IF you do), you are given a rating based on the number of days elapsed, and you can go back to sleeping in past sunrise every morning.

MACHINE HEAD

BY SPYDER



ARENA OF



OCTOS

BY AL JOHNSTON & STEVE D. KROPINAK

"Arena of Octos" is a graphic fantasy-conflict game for a 16K S-80, or a 32K Apple with Applesoft in ROM.

This pair of programs is unique. Normally, when we publish two or more listings of a given program for different computer systems, they consist of one "original" plus one or two "translations." Not so with "Arena of Octos." In this case there are two originals: The authors wrote them independently, after first discussing and agreeing upon the basic concept of the game.

You are the captain of a star cruiser which has been captured in an area of space claimed by the Octon Empire. The Octons are an aggressive race of creatures who evolved with eight fingers; thus the number "8" figures significantly in their culture (as does "10" in human cultures).

Octons hold individual courage and fighting ability in high regard. Therefore, the Council of Eight (ruling body of the planet Octos) has granted you the opportunity to free yourself

and your crew, by accepting "The Octon Challenge." In an eight-sided arena, armed with only sword and shield, you will battle Octon warriors in a series of increasingly difficult contests of strength and strategy.

Two hazards are in the arena: a firepit in the center, into which a combatant may fall; and eight stones, which may trip him. Each fighter's strength is measured in "strength points." He is defeated when all his strength points have been lost, either from an opponent's blows or from the hazards. If you are able to defeat all the Octons, you will be freed; if you are defeated by any of them, you and your crew will remain Octon slaves.

There are two "species" of Octons (depending upon which type of computer you are using), with slightly different characteristics and rules for formal combat.

APPLE VERSION

The Apple version utilizes Hi-Res graphics and sound. It requires 32K RAM, and Applesoft in ROM. All ac-

tion is controlled by game paddle 0: Turning the knob moves the selection cursor, and pressing the button makes the actual selection.

Octons are about the same size as humans, so all combatants begin with the same strength (16) and about the same "speed of action." A combatant may lose strength points in three ways:

1. Opponent's hit: A hit results in a loss of one point.

2. Falling down: A fall results in a loss of one point, and leaves the combatant lying down, undefended. Moving diagonally across a stone gives a 50% chance of tripping. A combatant who pushes another has a 20% chance of slipping; if the one pushed is already down, the pusher always slips. A combatant who is pushed may stumble; the better defended he is, the less will be his chance of stumbling.

3. Moving into the firepit: A combatant who trips, walks, or is pushed into the central firepit will lose a random number of points, and will then fall randomly outside the pit, lying down and undefended.

continued on next page

A combatant's speed is measured by his "action index." Your action index is 4 in the first contest, meaning that you may take four actions before the Octon's turn. For each action you may choose to move, strike, or defend.

1. Move. You may move in any of eight directions, or hold your present position. Choosing the move option will allow you to indicate the direction of your movement on a circular movement indicator on the screen. (Positioning the cursor at the center of this indicator means "hold position.") If you are lying down, you **MUST** move to "hold position" in order to get up; you cannot move to a new position until you do, although you may strike or defend while down. When a combatant moves into another's position, the second is pushed; the pushed combatant will not necessarily move directly opposite to the push.

2. Strike. A combatant must be within one move of an opponent to make a hit. If you are fighting two Octons, strength losses are subtracted from the closer one; if they are both one move away, losses are subtracted from the one with the higher action index (even if that Octon has already been defeated). The chance of hitting depends upon how well the opponent is defended. If you are defended when you strike, your status changes to undefended (or attack).

3. Defend. A combatant's defense is "+" (fully defended), "0" (partially defended), or "-" (undefended). When the "defend" action is taken, defense is set to "+". If an opponent hits (90% chance), no strength is lost, but the defense status decreases to "0". At "0" defense a good swing might be avoided by dodging (50% chance), but the defense status then drops to "-". At "-" defense, there is only a 10% chance of the opponent missing.

If you win a contest your action index increases, as does that of your opponent(s). After the first two contests you must fight two Octons at a time. If you defeat eight Octon warriors you win The Octon Challenge and will be freed.

The Apple species of Octons are aggressive, but not foolhardy. They will usually try to move toward you to attack. However, they will not purposefully walk through the firepit to get to you, and will try to end their turn defended, one move away from you. They will take a chance of crossing stones to reach you; and, if you are fighting two Octons, they won't hesitate to push each other out of the way to get to you. They usually won't

try to push you, however, unless you are near the firepit.

A few special notes on the Apple version should be observed. After typing in the program, **DO NOT RUN** it before you **SAVE** it! The first time you **RUN** the program, the shape tables and sound routine are **POKEd** into memory above the Hi-Res graphics map. Since the lines which do this are then no longer needed, they are then **DELETED** so that the program will occupy less space and fit into memory (just barely) under the Hi-Res area. Since the **DELETE** command stops program execution, you must then type **RUN** again to get on with it. After the first game ends (by either winning or losing The Octon Challenge), you need only type **RUN** once, since the Machine Language routines are still in memory and the program lines which do the **POKEing** are gone.

When you first type in the program, then, check lines 5010-5210 very carefully for errors; then **SAVE** the program; and then try a **RUN**. On the slim chance that you entered the whole thing without an error, your **SAVED** copy will be correct. If you do get an error along the way, you will have to **re-LOAD** the complete program before making corrections, then **re-SAVE** it, and then try **RUNning** again.

By the way, if you try any modifications to the program, be aware that you have only 43 free bytes to work with before you start overlapping the Hi-Res screen. If this happens, you will find the last characters or lines of the program disappearing after it is **RUN**, and it will have to be shortened.

S-80 VERSION

The S-80 version requires less than 16K of memory. Action is controlled through the keyboard, and optionally through a numeric keypad. The play is similar to that described above for the Apple, with a few differences.

You begin combat by facing one Octon, then two, then three, and eventually eight. Prior to actual combat, you are allowed a practice session if you choose. Each Octon begins with eight strength points and an action index of eight, whereas you have a strength rating and an action index of ten. During each round of the combat you may issue ten commands, and each of the Octons eight.

Commands are issued from the keyboard. When you want to move, you may use either a numeric keypad or a square arrangement of keys on the normal keyboard to specify one of eight directions:

- 1 or M: down-left.
- 2 or ,: down.
- 3 or .: down-right.
- 4 or J: left.
- 6 or L: right.
- 7 or U: up-left.
- 8 or I: up.
- 9 or O: up-right.

You may also swing your sword by pressing "S", raise your shield by pressing "R", or pass (and forfeit your remaining moves) by pressing "P".

S-80 Octons are smaller than humans, so they won't try to push you; however, you can still push them. If pushed across a stone, against another Octon, or against the arena wall, the Octon will have a greater chance of being hit. Their attack strategy is generally to swing, move, and swing again. The firepit does not frighten them, if it is the closest route to the human. Octons are, in fact, fearless — until the first one is removed from the arena. When this happens, the weakest of them will usually retreat and try to climb the arena wall, while the stronger ones fight the human.

Unlike some other programs, you cannot figure out an easy way to win by analyzing the code. The play will not be the same twice, even if the same moves are entered. The author himself spent a great deal of time fighting Octons until he successfully met the Octon challenge, and was freed from the arena long enough to send the program to **SoftSide**.

PROGRAMMING NOTE

Rarely do we receive programs with such extensive documentation as the authors supplied with "Arena of Octos." The listings which follow should serve as excellent examples for other programmers who send in submissions for publication.

APPLE VARIABLES

Unless otherwise noted, subscripts for array variables are keyed as follows: 0 = center of arena; 1 = human; 2-3 = Octons; 4-11 = stones; and 12 = new position.

- A: Actions left in combatant's turn.
- A(1-3): Action index of combatant (4-8).
- AN: Angle.
- B: General use.
- C: Combatant undergoing some process (drawing, decrementing strength, etc.)
- C\$(0): String for making random Octon names.
- C\$(1-3): Combatants' names.

CA: Combatant currently taking actions.
 CO: Contest number.
 CT: Combatant (or stone) whose position has been moved into (or crossed).
 D: Decision (move/strike/defend, or direction to move).
 D(1-3): Defense of combatant (+ 1, 0, or -1).
 DF(1-3): Flag for defeated combatant (+1 if defeated, 0 if not).
 E,F,G,H: Corners of octagons (pit and arena walls).
 K,M: Temporary value of combatant, or position when checking relative position.
 L,L1,L2,L3,L4: General use loop variables.
 L(1-3): Loss to be subtracted from combatant's strength.
 MD: Move direction for Octon (+ 1 = toward human, -1 = away).
 N: Number of combatants in contest (also, number of stones in initialization).
 ND: Total number of Octons defeated.
 P,P1,P2: Random numbers.

R(1-3): Rotation of combatant's shape.
 RS,R1,R2,R3: Square of radius between positions K and M.
 RT: Temporary radius-squared value.
 S: Shape (1 = human, 2 = Octon).
 SP(1-3): Shield position (puts combatant's shield on left, middle, or right).
 ST(1-3): Remaining strength of combatant.
 X,Y: General use position variables.
 X(0-12),Y(0-12): Current position.
 XD,YD,X1,Y1,X2,Y2,X3,Y3: Difference in horizontal and vertical positions of K and M.
 XN,YN: New position.

DD: Practice flag.
 DS: Distance from Octon to human.
 DX,DY: Difference between Octon and human positions.
 F\$(25): Fire string.
 FG: General utility flag.
 G\$: Plural string.
 GG: Message variable.
 HD: Human-shielded flag.
 HL: Number of human moves left.
 HM: Number of human moves (10).
 HS: Human strength.
 HT: "Human's turn" flag.
 HX,HY: Human's position.
 I,J,K,L,M,N: Iteration variables.
 LF: Number of Octons left.
 M\$: Message string.
 MM: Message number.
 MV: Move (1 - 4 or 6 - 9).
 NC: Number of Octons.
 NX,NY: New location.
 PX,PY: Present location.
 P\$: Player string.
 RN: Random number.
 SB: "Struck before?" flag.
 WK: Weakest Octon's strength.
 X1,Y1: Temporary X and Y locations.
 X,Y: Graphics variables.

S-80 VARIABLES

AN: Out-of-arena flag.
 A\$: INKEY\$ variable.
 BD(14,14): Board array. (Defines out-of-arena, pit, and stone positions.)
 C(8,8): Octon array. (Contains Octon position and status: strength, number of moves left, etc.)
 CL: Number of Octon moves left.
 CX,CY: Octon's present position.

Apple Version

Go to routine to start game.

```
5 LOMEM: 17000: BOTO 5000
```

Generate random numbers.

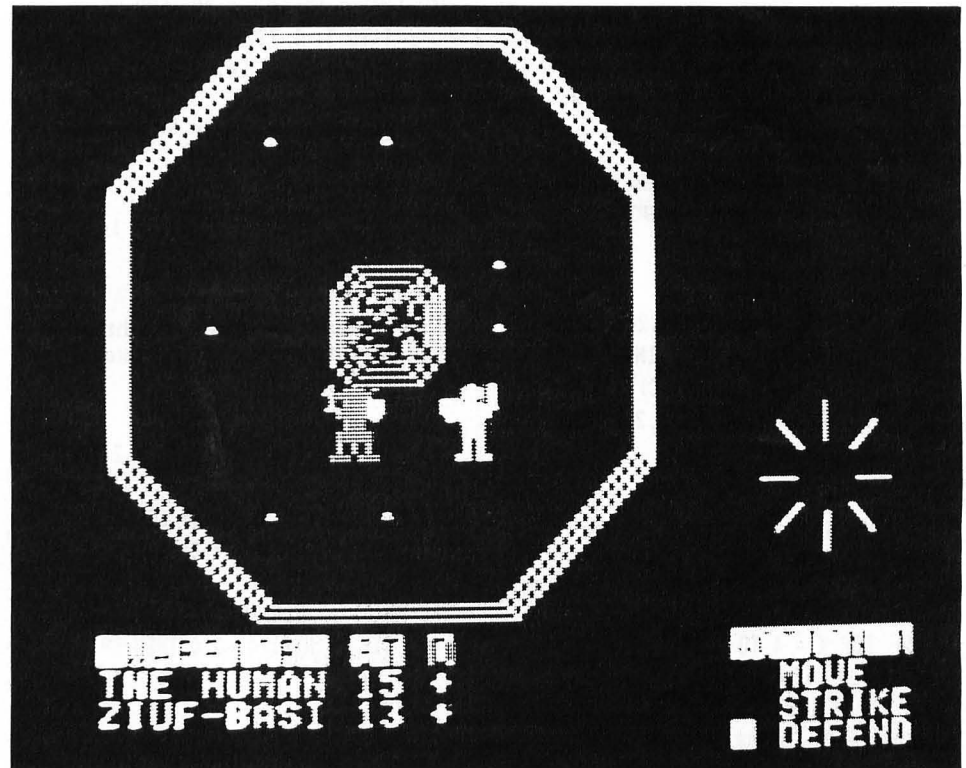
```
10 P = RND (1) * 100:P1 = INT (
  RND (1) * 3) - 1:P2 = INT
  ( RND (1) * 3) - 1: RETURN
```

Make coals in firepit glow.

```
20 GOSUB 10: HCOLOR= 5: IF P < 5
  0 THEN HCOLOR= 0
25 X = P1 * RND (1) * 10 + 80:Y =
  P2 * RND (1) * 10 + 80: HPLLOT
  X,Y: RETURN
```

Print combatant's strength and defense status; decrement strength if needed. (Pokes set text window to 16 spaces at left of screen.)

```
30 POKE 32,0: POKE 33,16: HOME :
  VTAB 21: INVERSE : PRINT "
  WARRIOR "; HTAB 11: PRINT "
  ST"; HTAB 14: PRINT "D": NORMAL
  : FOR L = 1 TO N: VTAB 21 +
```



```
L: PRINT C$(L); IF ST(L) <
  10 AND ST(L) > - 1 THEN PRINT
  " ";
35 PRINT " ";ST(L);" "; IF D(L)
  = 1 THEN PRINT "+";
40 IF D(L) = 0 THEN PRINT "0";
45 IF D(L) = - 1 THEN PRINT "-
```

```
";
50 IF L < 3 THEN PRINT ""
60 NEXT : FOR L = 1 TO N: IF L(L)
  ) > 0 THEN ST(L) = ST(L) - 1
  :L(L) = L(L) - 1: FOR L1 = 1
  TO 4: & T160,10: & T240,20:
  NEXT L1: GOTO 30
```

continued on next page

continued from previous page

65 NEXT : RETURN

Set combatant's defense to fully defended (+).

70 HOME : PRINT C\$(C);" DEFENDS.
": IF D(C) = 1 THEN 4000

75 GOSUB 220;D(C) = 1: GOSUB 250
: GOTO 4000

Check whether position M is the same as (or near) any of N other positions. CT returns with the subscript value of whatever is in the same position as M, or with 0 if nothing is in the same position.

80 CT = 0: FOR K = 1 TO N: IF K =
M THEN NEXT : RETURN

85 GOSUB 100: IF RS < 180 THEN C
T = K:K = N: NEXT : RETURN

90 NEXT : RETURN

Determine relative distances of position K from position M.

100 X(12) = XM:Y(12) = YM:XD = X(
K) - X(M):YD = Y(K) - Y(M):R
S = XD ^ 2 + YD ^ 2: RETURN

Print action heading. (Pokes set text window to 20 spaces at right of screen.)

110 POKE 33,20: POKE 32,20: HOME
: VTAB 21: HTAB 7: INVERSE :
PRINT "ACTION ";A(C) - A +
1: NORMAL : POKE 34,21: HOME
: RETURN

Check whether a combatant has crossed a stone; if so, determine whether a fall occurs.

150 FOR K = 4 TO 11:M = C: GOSUB
100: IF RS > 200 THEN NEXT
: RETURN

160 M = 12: GOSUB 100: IF RS > 20
0 THEN NEXT : RETURN

170 B = K:K = C: GOSUB 100:K = B:
IF RS < 400 THEN NEXT : RETURN

180 K = 11: NEXT : GOSUB 10: GOSUB
200: IF P < 50 THEN R(C) = 3
2 + 16 * SGN (XD):D(C) = -
1:L(C) = 1: PRINT C\$(C);" TR
IPS": PRINT "OVER A STONE!"

190 RETURN

Lines 200-230 erase a combatant's shape:

Determine which shape to erase.

200 ROT= R(C):S = 1: IF C > 1 THEN
S = 2

Erase combatant.

210 HCOLOR= 0: DRAW S AT X(C),Y(
C)

220 HCOLOR= 0

Erase (or draw) shield and sword.

230 DRAW 3 AT X(C) + 6 * SP(C),Y
(C) - 3: DRAW 4 AT X(C) - 6 *
SP(C),Y(C) - 3: ROT= 0: RETURN

Lines 240-290 draw a combatant's shape:

Set combatant's position to new position; make movement sound.

240 X(C) = XN:Y(C) = YN: FOR L3 =
1 TO 3: & T255,2: FOR L4 = 1
TO 40: NEXT L4,L3

Determine where to draw sword and shield (left, middle, right).

250 M = C: IF C > 1 THEN K = 1: GOSUB
100: GOTO 280

If combatant is human and fighting 2 Octons, determine which is closer.

260 K = 2: GOSUB 100: IF N = 2 THEN
280

270 RT = RS:K = 3: GOSUB 100: IF
RS > RT THEN K = 2: GOSUB 10
0

280 SP(C) = SGN (D(C)) * SGN (X
D): IF R(C) < > 0 THEN SP(C)
) = 0

Determine which shape to draw.

285 ROT= R(C):S = 1: IF C > 1 THEN
S = 2

Draw combatant.

290 HCOLOR= 1 + 5 * (2 - S): DRAW
S AT X(C),Y(C): HCOLOR= 3: GOTO
230

Lines 300-390 draw a sword swing and determine the results:

If combatant is defending, set defense to attack position (-).

300 HOME : PRINT C\$(C);" STRIKES
...": IF D(C) < > - 1 THEN
D(C) = - 1: GOSUB 220: GOSUB
250

Draw sword down-swing.

310 FOR L = 1 TO 3: GOSUB 370: NEXT
: GOSUB 390: IF C > 1 THEN K
= 1: GOTO 330

Check whether Octon is close enough to hit.

315 M = 1: FOR K = 2 TO N: GOSUB
100: IF RS < 600 THEN CT = K
:K = N: NEXT :K = CT: GOTO 3
30

Octon is too far away to hit.

320 NEXT : FOR L = 1 TO 500: NEXT
: PRINT "AND MISSES!": PRINT
"A WILD SWING!": GOTO 360

If opponent is fully defended, check for a deflection (90% chance); make deflection sound.

330 GOSUB 10: IF D(K) = 1 AND P >
10 THEN PRINT C\$(K);" DEFLE
CTS THE BLOW.":D(K) = 0: &
T150,3: & T255,3: GOTO 360

If opponent is half-defended, check for a dodge (50% chance).

335 IF D(K) = 0 AND P > 50 THEN
PRINT C\$(K);" DODGES THE
BLOW!":D(K) = - 1: GOTO 3
60

Check for a hit (90% chance).

340 IF P > 10 THEN 350
345 PRINT "AND MISSES...": GOTO
360

Hit sound.

350 FOR L1 = 1 TO 4: & T200,2: FOR
L2 = 1 TO 20: NEXT L2,L1: PRINT
"AND HITS ";C\$(K);"!":L(K) =
1

Draw sword upswing.

```
360 FOR L = 2 TO 0 STEP - 1: GOSUB
370: NEXT : GOSUB 390: GOTO
4000
```

Erase sword and rotate it in swing.

```
370 HCOLOR= 0: GOSUB 380: ROT= 6
4 - L * 8 * SP(C): HCOLOR= 3
```

Draw (or erase) sword.

```
380 DRAW 4 AT X(C) - 6 * SP(C),Y
(C) - 3: RETURN
```

Sword swing sound.

```
390 FOR L1 = 0 TO 48 STEP 3: & T
L1,2: NEXT L1: RETURN
```

Lines 400-460 determine results of combatant's move:

If combatant is standing and holds position, print message and return.

```
400 HOME : PRINT C$(C); IF D =
0 AND R(C) = 0 THEN PRINT "
HOLDS POSITION.": GOTO
4000
```

If combatant is lying down and holds position, stand him up.

```
410 IF D = 0 AND R(C) < > 0 THEN
PRINT " GETS UP.": GOSUB 20
0:R(C) = 0:YN = X(C):Y = Y(
C): GOTO 460
```

If combatant is lying down and tried to move, print message and return.

```
420 PRINT " MOVES...": IF R(C) <
> 0 AND D < > 0 THEN PRINT
"BUT CAN'T GET UP.": FOR L =
1 TO 800: NEXT : GOTO 4000
```

Check whether combatant tried to jump out of arena; if so, draw jump.

```
430 M = 12:K = 0: GOSUB 100: IF R
S > 4000 THEN PRINT "AND TR
IES TO JUMP OUT OF THE ARE
NA!": GOSUB 200:X = X(C):Y =
Y(C): GOSUB 240: GOSUB 200:X
M = X:YN = Y: GOSUB 240: GOTO
4000
```

Check whether combatant moved into firepit.

```
440 IF RS < 500 THEN PRINT "AND
FALLS": PRINT "INTO THE FIR
EPIT!": GOSUB 200: GOSUB 60
0: GOTO 4000
```

Check for a push.

```
450 M = 12: GOSUB 80: IF CT < >
0 THEN 500
```

Check for crossing a stone; draw move.

```
460 GOSUB 150: GOSUB 200: GOSUB
240: GOTO 4000
```

Lines 500-590 determine the results of a push:

If a combatant tries to push another who is already down, the one pushing trips.

```
500 M = C:K = CT: GOSUB 100:X1 =
XD:Y1 = YD: IF R(CT) < > 0 THEN
PRINT C$(C);" TRIPS": PRINT
"OVER ";C$(CT);".": GOSUB 20
0: GOSUB 590:X(C) = XM:Y(C) =
YN: GOTO 550
```

Determines whether pushing combatant slips (20% chance).

```
510 PRINT C$(C);" PUSHES": PRINT
C$(CT);"...": GOSUB 10: IF P
< 20 THEN GOSUB 200: GOSUB
590:XM = X(C):YN = Y(C): PRINT
C$(C);" SLIPS!": GOSUB 240:
GOTO 4000
```

Determine whether pushed combatant stumbles (25% chance if fully defended, 50% if half-defended, 75% if undefended).

```
520 C = CT: GOSUB 200:C = CA: GOSUB
200: GOSUB 240:C = CT: IF P -
D(C) * 25 > 50 THEN GOSUB 5
90: HOME : PRINT C$(C);" STU
MBLES!"
```

Determine new position of pushed combatant.

```
530 YN = Y(C) + 16 * SGN (Y1):XM
= X(C) + 16 * SGN (X1): IF
X1 < 8 THEN XM = X(C)
```

Check whether new position is out of arena.

```
540 K = 12:M = 0: GOSUB 100:RT =
RS: IF RS < 4000 THEN 560
```

Pick an alternate new position, randomly about old position.

```
550 GOSUB 10:XM = X(C) + 16 * P1
:YN = Y(C) + 16 * P2: GOTO 5
40
```

Check whether new position is in another combatant's position; if so, select a new one.

```
560 M = 12: GOSUB 80: IF CT < >
0 THEN 550
```

Check if new position is in firepit.

```
570 IF RT < 500 THEN PRINT C$(C
);" FALLS": PRINT "INTO THE
FIREPIT!": GOSUB 600
```

Draw combatant in new position.

```
580 GOSUB 240:C = CA: GOTO 4000
```

Rotate shape, set defense to undefended (-), and set strength loss to 1 for combatant who falls.

```
590 R(C) = 32 - 16 * SGN (XD):D(
C) = - 1:L(C) = 1: RETURN
```

Lines 600-630 are the firepit subroutine:

Put combatant in firepit, make burning sounds, set loss to random number.

```
600 L(C) = L(C) + 1:B = B + 1: IF
B > 4 THEN B = 1
```

```
610 R(C) = 16 * B: GOSUB 10:XM =
80 + B * P1 + INT (C / 2):Y
N = 80 + B * P2: GOSUB 240: &
T1 + B,15: FOR L1 = 1 TO P /
20: & T255,2: FOR L2 = 1 TO
P / 10: NEXT L2,L1: GOSUB 20
0: IF P1 = 0 OR P2 = 0 THEN
600
```

```
620 XM = (24 + INT (C / 2)) * P1
+ 80:YN = 24 * P2 + 80:M =
12: GOSUB 80: IF CT < > 0 THEN
600
```

continued on next page

continued from previous page

Toss combatant out of pit, lying down and undefended.

```
630 GOSUB 200:R(C) = 32 + 16 * P
    1:D(C) = - 1: GOSUB 240: RETURN
```

Draw arena walls and firepit.

```
800 HGR
```

```
810 E = 8:F = 48:G = 112:H = 152:
    HCOLOR= 6: GOSUB 820:E = 69
    :F = 75:G = 85:H = 91: HCOLOR=
    5
```

```
820 FOR L = 1 TO 3: HPLLOT F,E TO
    G,E TO H,F TO H,G TO G,H TO
    F,H TO E,G TO E,F TO F,E: E =
    E - 2:F = F - 2:G = G + 2:H =
    H + 2: NEXT L: RETURN
```

Lines 1000-1800 initialize new contest:

Draw arena and pit; start loop for stone placement.

```
1000 GOSUB 800: HCOLOR= 3: ROT=
    0:B = N:N = 4: FOR M = 4 TO
    11
```

Pick random stone positions; check if overlapping another stone or outside arena.

```
1100 GOSUB 10:X(M) = INT ( RND
    (1) * 4) * 16 * P1 + 80:Y(M)
    = INT ( RND (1) * 4) * 16 *
    P2 + 80: GOSUB 80: IF CT < >
    0 THEN 1100
```

```
1200 K = 0: GOSUB 100: IF RS > 40
    00 OR RS < 500 THEN 1100
```

Plop the stones in the arena.

```
1300 DRAW 6 AT X(M),Y(M):N = N +
    1: & T50,5: & T255,5: NEXT :
    N = 8: FOR M = 1 TO N
```

Pick random positions for combatants.

```
1400 GOSUB 10: IF P1 = 0 OR P2 =
    0 THEN 1400
```

```
1500 X(M) = ( INT ( RND (1) * 5) *
    16 + 8 + INT ( M / 2)) * P1 +
    80:Y(M) = ( INT ( RND (1) *
    5) * 16 + 8) * P2 + 80: GOSUB
    80: IF CT < > 0 THEN 1400
```

```
1600 K = 0: GOSUB 100: IF RS > 40
    00 OR RS < 500 THEN 1400
```

Draw combatant, set strength to full (16), defense to undefended (-); set turn to human's.

```
1700 NEXT : FOR C = 1 TO N:D(C) =
    - 1:ST(C) = 16:R(C) = 0:XM =
    X(C):YN = Y(C):DF(C) = 0: GOSUB
    240: NEXT :C = 1:XM = X(1):Y
    N = Y(1)
```

Draw direction indicator.

```
1800 HCOLOR= 3: FOR L = 0 TO 7:A
    N = L * 3.14 / 4: HPLLOT 210 +
    10 * COS (AN),120 + 10 * SIN
    (AN) TO 210 + 20 * COS (AN)
    ,120 + 20 * SIN (AN): NEXT
    : GOSUB 30
```

Lines 2000-2160 handle human's move:

Use paddle 0 to move decision cursor and choose option. (Peek checks for paddle button press.)

```
2000 GOSUB 110: HTAB 9: PRINT "M
    OVE": HTAB 9: PRINT "STRIKE"
    : HTAB 9: PRINT "DEFEND":
    2010 D = INT ( PDL (0) / 90) + 1
    : VTAB 22: HTAB D * 40 - 33:
    INVERSE : PRINT " "; IF PEEK
    (49249) > 127 THEN 2030
    2020 NORMAL : GOSUB 20: VTAB 22:
    HTAB D * 40 - 33: PRINT " "
    ; GOTO 2010
    2030 NORMAL : FOR L = 1 TO 1000:
    NEXT : DN D GOTO 2100,300,7
    0
```

Human decided to move; now choose direction.

```
2100 HOME : VTAB 22: HTAB 6: PRINT
    "DIRECTION?"
    2110 HCOLOR= 3:D = INT ( PDL (0)
    ) / 29: IF D = 0 THEN X = 2
    10:Y = 120: GOTO 2130
    2120 AN = D * 3.14 / 4:X = 210 +
    25 * COS (AN):Y = 120 + 25 *
    SIN (AN)
    2130 DRAW 5 AT X,Y: IF PEEK (49
    249) > 127 THEN 2150
    2140 XDRAW 5 AT X,Y: GOSUB 20: GOTO
    2110
    2150 FOR L = 1 TO 500: NEXT : XDRAW
```

```
5 AT X,Y: IF D = 0 THEN 400
2160 XM = X(1) + 16 * SGN ( INT
    ( COS (AN) + .5)):YN = Y(1) +
    16 * SGN ( INT ( SIN (AN) +
    .5)): GOTO 400
```

Lines 3000-3150 handle Octon's move:

If Octon is defeated, go to next combatant's turn.

```
3000 IF DF(C) = 1 THEN A = 1: GOTO
    4000
```

If Octon is down, he gets up.

```
3010 GOSUB 110:MD = 1:D = 1: IF
    R(C) < > 0 THEN D = 0: GOTO
    400
```

Find relative positions of human, firepit, and Octon.

```
3020 M = C:K = 1: GOSUB 100:R1 =
    RS:X1 = XD:Y1 = YD:M = 0: GOSUB
    100:R2 = RS:X2 = XD:Y2 = YD:
    K = C: GOSUB 100: IF (A > 3 AND
    R1 > 600) OR (A > 1 AND R1 >
    2200) OR (A < 3 AND R(1) = 0
    AND D(1) < 1 AND R2 < 1500)
    THEN 3100
    3030 IF A > 1 AND R1 > 600 THEN
    D = 0: GOTO 400
    3040 IF A = 1 AND R1 > 600 THEN
    70
    3050 IF A > 2 OR (D(1) < 1 AND (
    (A = 2 AND ST(1) < 3) OR (A =
    1 AND ST(1) = 1))) THEN 300
    3060 IF A = 2 THEN MD = - 1: GOTO
    3100
    3070 GOTO 70
```

Octon decided to move; determine new direction. Check for move into pit.

```
3100 IF ABS (X1) < 8 THEN X1 =
    0
    3110 XM = X(C) + 16 * MD * SGN (
    X1):YN = Y(C) + 16 * MD * SGN
    (Y1):K = 12:X3 = XD:Y3 = YD:
    GOSUB 100: IF RS > 500 THEN
    400
    3120 IF SGN (Y1) = 0 THEN YN =
    YN + 16 * SGN (Y3): GOTO 40
    0
    3130 IF SGN (X1) = 0 THEN XN =
    XN + 16 * SGN (X3): GOTO 40
    0
```

```

3140 IF ABS (Y3) > 16 THEN YN =
      Y(C): GOTO 400
3150 XN = X(C): GOTO 400

```

Lines 4000-4050 handle results at the end of an action:

Print new strength and defense status; redraw combatants, arena, and stones; check for a defeat.

```

4000 GOSUB 30: FOR C = 1 TO N: GOSUB
      220: GOSUB 250: NEXT C: CA =
      CA: GOSUB 810: HCOLOR= 3: FOR
      L = 4 TO 11: DRAW 6 AT X(L),
      Y(L): NEXT L: FOR L = 1 TO N:
      IF ST(L) < 1 AND DF(L) = 0 THEN
      B = L: L = 3: NEXT L: GOTO 410
      0
4010 NEXT

```

Decrement actions remaining; check whether combatant has any left.

```

4020 A = A - 1: IF A < 0 THEN
      4030

```

Sets turn for next combatant.

```

4030 C = C + 1: IF C > N THEN C =
      1
4040 A = A(C)

```

Go to routine for human's or Octon's move.

```

4050 CA = C: ON C GOTO 2000,3000,
      3000

```

Combatant has been defeated. Set defeat flag, draw defeated contestant, make sound; if Octon, check if all Octons are defeated.

```

4100 DF(B) = 1: CA = C: C = B: GOSUB
      200: R(B) = 16: GOSUB 250: C =
      CA: GOSUB 110: PRINT C$(B): PRINT
      "IS DEFEATED!": FOR L = 1 TO
      160: & TL,5: NEXT L: IF B > 1
      THEN 4300

```

Human is defeated.

```

4200 TEXT : HOME : VTAB 4: PRINT
      " YOU HAVE LOST THE OCTON C
      HALLENGE, AND MUST NOW WORK I
      N THE OCTAL PALACE FOR SCR
      APS OF FOOD...": PRINT : PRINT
      " UNTIL YOU ARE READY TO TR
      Y AGAIN!": FOR L = 1 TO 160:
      & T RND (1) & 250,4: NEXT L:
      END

```

Increment number of Octons defeated.

```

4300 ND = ND + 1: IF ND < 2 &
      CO - 2 AND CO < 1 THEN 40
      00

```

Contest won by human.

```

4400 TEXT : HOME : VTAB 8: PRINT
      " YOU HAVE DEFEATED ";ND;"
      OF THE 8 WARRIORS.": FOR L =
      1 TO ND: & T160 - L & 10,40:
      & T160,40: NEXT L

```

If all 8 Octons have not been defeated, increment action index of human and begin next contest.

```

4500 IF ND < 8 THEN A(1) = A(
      1) + 1: PRINT : PRINT " EXP
      ERIENCE HAS IMPROVED YOUR SK
      ILL: YOU NOW HAVE AN ACTIO
      N INDEX OF ";A(1);".": GOTO
      6000

```

Octon challenge has been won; end program.

```

4600 VTAB 16: PRINT " YOU HAVE
      WON THE OCTON CHALLENGE, AND
      ARE FREE TO GO!": END

```

Initialization.

```

5000 TEXT : HOME
5010 VTAB 8: PRINT TAB( 12)"THE
      ARENA OF OCTOS": PRINT : PRINT
      TAB( 11)"BY STEVE D. KROPIN
      AK": PRINT : PRINT TAB( 17)
      "(6/1/81)"
5020 VTAB 16: PRINT TAB( 11)"32
      K APPLESOFT IN ROM"

```

Poke in Machine Language routines upon running program for the first time.

```

5030 FOR L = 16640 TO 16925: READ
      B: POKE L,B: NEXT L
5040 POKE 232,0: POKE 233,65: POKE
      1013,76: POKE 1014,220: POKE
      1015,65
5050 VTAB 20: PRINT "TYPE 'RUN'
      TO BEGIN.": DEL 5010,5210

```

Data (5100 - shape table data; 5110-5130 - human shape; 5140-5170 - Octon shape; 5180 - shield, sword, direction marker, and stones; 5190-5210 - &T sound routine).

```

5100 DATA 6,0,14,0,85,0,182,0,19
      8,0,205,0,215,0

```

```

5110 DATA 36,36,36,36,60,45,21,6
      3,63,46,45,53,63,63,14,45,21
      ,63,63,223,19,45,45,45,45

```

```

5120 DATA 45,45,30,63,63,63,63,6
      3,14,45,45,45,45,30,63,63,63
      ,14,45,45,62,63,55,45,45

```

```

5130 DATA 62,63,55,109,53,255,55
      ,109,53,255,255,42,45,13,45,
      53,63,255,63,63,0

```

```

5140 DATA 36,36,36,36,108,73,17,
      223,59,63,255,27,14,45,45,45
      ,45,45,30,63,63,63,63,14,45

```

```

5150 DATA 45,45,30,63,63,223,19,
      45,45,45,45,45,45,30,63,63,6
      3,63,63,14,45,45,45,45,30

```

```

5160 DATA 63,63,63,14,45,45,62,6
      3,255,42,45,45,45,53,63,63,6
      3,63,46,45,45,45,53,223,251

```

```

5170 DATA 27,110,9,77,49,223,251
      ,27,159,45,45,77,41,45,53,63
      ,63,223,59,63,63,0

```

```

5180 DATA 63,36,45,45,62,63,55,4
      5,45,62,63,119,45,30,7,0,5,6
      3,12,36,36,36,0,18,63,32,100
      ,45,21,54,30,7,0,21,63,7,0

```

```

5190 DATA 0,201,84,208,15,32,177
      ,0,32,248,230,138,72,32,183,
      0,201,44,240,3,76,201,222,32
      ,177

```

```

5200 DATA 0,32,248,230,104,134,3
      ,134,1,133,0,170,160,1,132,2
      ,173,48,192,136,208,4,198,1,
      240

```

continued on next page

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 Tardus, and commissioned you to eliminate the evil "Master." Your resources include clones who will fight for you, the formidable CRASER weapons of the Tardus, 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.

And once you find The Master can you destroy him?

(Time Lord is based on Dr. Who of PBS fame.)

Apple Integer
BASIC, Disk, 48K
..... \$29.95



continued from previous page

```
5210 DATA 7,202,208,246,166,0,20
      8,239,165,3,133,1,198,2,208,
      241,96
```

Game initialization after deleting of DATA lines (to reduce program size below Hi-Res screen memory).

```
5500 DIM X(12),Y(12):X(0) = 80:Y
      (0) = 80: SCALE= 1:A(1) = 4:
      C$(0) = "BAKUFIDONAGORELIJIM
      UPORASITOVEZI":C$(1) = "THE
      HUMAN"
```

```
5600 VTAB 4: PRINT " YOUR STAR
      CRUISER HAS BEEN CAPTURED
      IN SPACE CLAIMED BY THE OCTO
      N EMPIRE.": PRINT
```

```
5700 PRINT " HOWEVER, THE COUNC
      IL OF 8 WILL GRANT YOU FREE
      DOM IF YOU PROVE YOUR STRENG
      TH AND COURAGE IN THE OCTON
      CHALLENGE: YOU MUST DEFEAT
      8 OCTON WARRIORS IN ARENA
      COMBAT."
```

```
5800 PRINT : PRINT " YOUR ACTIO
      N INDEX IS 4."
```

Initialization of a new contest: Set first turn to human's, actions remaining to human's action index; determine number of Octen opponents; make up random names for Octons; print message to begin.

```
6000 CA = 1:A = A(1):CD = CD + 1:
      N = 2: IF CD > 2 THEN N = 3
6100 FOR L = 2 TO N:C$(L) = "": FOR
      L1 = 1 TO 4:C$(L) = C$(L) +
      MID$(C$(0), INT ( RND (1) *
      31 + 1),2): IF L1 = 2 THEN C
      $(L) = C$(L) + "-"
6200 NEXT :A(L) = CD + 5 - L: NEXT
```

```
6300 VTAB 16: PRINT " IN CONTES
      T ";CD;", YOU WILL BATTLE": FOR
      L = 2 TO N: PRINT C$(L);", W
      HO HAS AN ACTION INDEX OF ";
      A(L);: IF N = 3 AND L = 2 THEN
      PRINT " "; SPC( 16);"AND"
6400 NEXT : PRINT ".": VTAB 22: PRINT
      TAB( 8)"PRESS BUTTON WHEN R
      EADY."
6500 IF PEEK (49249) > 127 THEN
      HOME : GOTO 1000
6600 GOTO 6500
```


S-80 Version

20 B0T01000

Location subroutine. Input: HX, HY is human position; CX, CY is Octon position. Output: MV is move; DS is distance.

```

30 DX=HX-CX: DY=HY-CY
40 IFDX>0THENIFDY>0THENDS=DY: MV=2: RETURNSEDS=-DY: MV=8: RETURN
50 IFDY>0THENIFDX>0THENDS=DX: MV=6: RETURNSEDS=-DX: MV=4: RETURN
60 DS=ABS(DX): IFDS<ABS(DY) THENDS=ABS(DY)
70 IFDY>0THENIFDX>0THENMV=3: RETURNSEMV=1: RETURN
80 IFDX>0THENMV=9: RETURNSEMV=7: RETURN
    
```

Check for Octon removed; set flag if so.

```

90 FG=0: IFC(I,4)<=0THENC(I,4)=0: PRINT@876+2*I, " "; PRINT@940+2*I, " "; FG=-1: RETURNSERETURN
    
```

Check for ALL Octons removed; if so, human wins.

```

100 LF=0: FORK=1T0NC: PRINT@468, F*(K+8); IFC(K,0) THENLF=LF+1: NEXTK ELSENEXTK
110 IFLF>0THENRETURNSE4000
    
```

Move direction subroutine. Input: MV is move; PX, PY is present location. Output: NX, NY is new location.

```

120 NX=PX: NY=PY: ONMVGOTO130,140,150,160,,170,180,190,200
130 NX=PX-1: NY=PY+1: RETURN
140 NY=PY+1: RETURN
150 NX=PX+1: NY=PY+1: RETURN
160 NX=PX-1: RETURN
170 NX=PX+1: RETURN
180 NX=PX-1: NY=PY-1: RETURN
190 NY=PY-1: RETURN
200 NX=PX+1: NY=PY-1: RETURN
    
```

Shield-raised subroutine.

```

210 PRINT@109, "SHIELD RAISED "; IFHTTHENPRINT@875, "Y"; HD=-1: RETURNSEPRINT@876+2*I, "Y"; C(I,5)=-1: RETURN
    
```

Shield-down subroutine.

```

220 PRINT@109, "SHIELD LOWERED"; IFHTTHENPRINT@875, "N"; HD=0: RETURNSEPRINT@876+2*I, "N"; C(I,5)=0: RETURN
230 IFHTTHENMM=HLELSEMM=C(I,7)
    
```

Message subroutine.

```

240 PRINT@107+64*MM, MM, M%; IFHTTHENRETURNSEFORK=1T03: FORJ=1T02
5: PRINT@468, F*(J); NEXTJ, K: RETURN
    
```

Change the strength display.

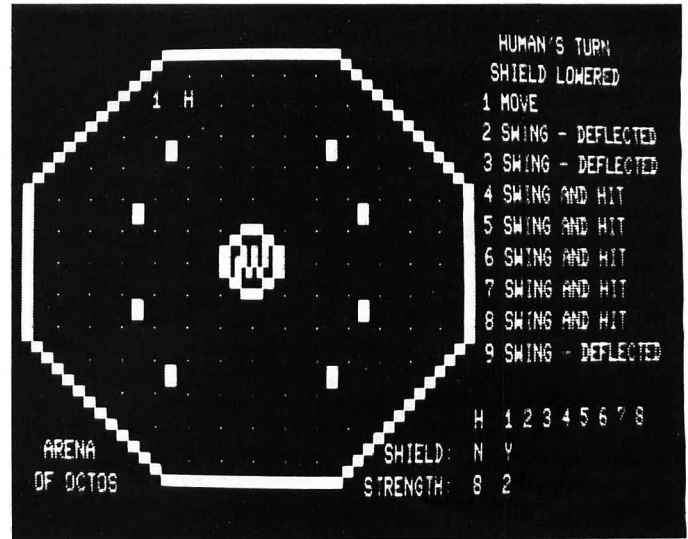
```

250 IFHTTHENPRINT@938, USING"##"; HS; RETURNSEPRINT@939+2*I, C(I,4); RETURN
    
```

Clear the message display.

```

260 FORL=171T0747STEP64: PRINT@L, STRING*(21,32); NEXTL: RETURN
    
```



Put the player in new position.

```

270 PRINT@640PY+30PX, ". "; PRINT@640NY+30NX, P%; RETURN
    
```

Out-of-arena? subroutine. Input: NX, NY is new (proposed) position. Output: AN is out-of-arena flag.

```

280 AN=0: IFBD(NX, NY)=-2THENAN=-1: RETURNSERETURN
    
```

Into pit? subroutine. Input: NX, NY is new (proposed) position. Output: FG is pit flag.

```

290 FG=0: IFBD(NX, NY)=-1THENFG=-1: RETURNSERETURN
    
```

Subroutine to check for stone. Input: PX, PY is present location; NX, NY is new location. Output: FG is "passed stone" flag.

```

300 FG=0: IFBD(NX, NY)>2 AND BD(NX, NY)=BD(PX, PY) THENFG=-1: RETURNSERETURN
    
```

In-the-pit subroutine. Input: P\$ is player; HT is human's-turn flag. Output: NX, NY is new location. Internal: J, K, FG

```

310 FG=0: RN=RND(8): ONRNGOTO320,330,340,350,360,370,380,390
320 NX=7: NY=5: GOTO400
330 NX=8: NY=6: GOTO400
340 NX=9: NY=7: GOTO400
350 NX=8: NY=8: GOTO400
360 NX=7: NY=9: GOTO400
370 NX=6: NY=8: GOTO400
380 NX=5: NY=7: GOTO400
390 NX=6: NY=6
400 IFHX=NXANDHY=NYTHEN310
410 FORK=1T03: FORN=1T025: PRINT@468, LEFT$(F*(N),1)+P$+RIGHT$(F*(N),1); NEXTN, K
420 FORJ=1T0NC: PRINT@468, LEFT$(F*(J),1)+P$+RIGHT$(F*(J),1); IFC(J,0)=0THEN440
430 IFC(J,1)=NXANDC(J,2)=NYTHEN310
    
```

continued on next page

continued from previous page

```
440 NEXTJ:IFHTTHEN460
450 C(I,1)=NX:C(I,2)=NY:C(I,4)=C(I,4)-1:GOSUB90:IFF6THENC(I,0)=0
:GOSUB100:RETURNELSEGOSUB220:GOSUB250:PRINT@469," ";:GOSUB270:RE
TURN
460 HX=NX:HY=NY:HS=HS-1:IFHS<=0THEN6000ELSEGOSUB220:GOSUB250:PRI
NT@469," ";:GOSUB270:RETURN
```

Program begins. Declaration and initialization. Display title page while loading arrays.

```
1000 RANDOM:CLEAR150:DEFINTA-Z:DIMC(8,8),BD(14,14),F*(25)
1010 CLS:PRINTCHR*(23);:PRINT@398,"ARENA OF OCTOS":PRINT@718,"BY
AL JOHNSTON":PRINT@782,"APRIL 25, 1981"
1020 FORY=0TO14:FORX=0TO14:BD(X,Y)=-2:NEXTX,Y
1030 FORY=1TO5:X1=6-Y:X2=8+Y:GOSUB1040:NEXTY:GOTO1050
1040 FORX=X1TOX2:BD(X,Y)=0:BD(X,14-Y)=0:NEXTX:RETURN
1050 FORY=6TO8:FORX=1TO13:BD(X,Y)=0:NEXTX,Y
1060 X=4:Y=3:GOSUB1070:X=9:GOSUB1070:Y=10:GOSUB1070:X=4:GOSUB107
0:GOTO1080
1070 BD(X,Y)=1:BD(X+1,Y+1)=1:BD(X+1,Y)=3:BD(X,Y+1)=3:RETURN
1080 X=3:Y=5:GOSUB1090:X=10:GOSUB1090:Y=8:GOSUB1090:X=3:GOSUB109
0:GOTO1100
1090 BD(X,Y)=5:BD(X+1,Y+1)=5:BD(X+1,Y)=4:BD(X,Y+1)=4:RETURN
1100 FORX=6TO8:BD(X,5)=2:BD(X,9)=2:NEXTX
1110 FORY=6TO8:FORX=5TO9:BD(X,Y)=2:NEXTX,Y
1120 BD(7,6)=-1:BD(6,7)=-1:BD(7,7)=-1:BD(8,7)=-1:BD(7,8)=-1
1130 FORI=1TO25:F*(I)=CHR*(128+RND(63))+CHR*(128+RND(63))+CHR*(1
28+RND(63)):NEXTI
```

Ask for instructions, practice session.

```
1140 CLS:PRINT@450,"DO YOU WANT INSTRUCTIONS?
1150 A$=INKEY$:IFA$=""THEN1150ELSEIFA$="Y"THEN5000ELSEIFA$("<"N"
HENI1150
1160 PRINT"DO YOU WANT A PRACTICE SESSION?
1170 A$=INKEY$:IFA$=""THEN1170ELSEIFA$="Y"THENDD=0ELSEIFA$="N"
THENDD=-1ELSE1170
1180 NC=0
1190 NC=NC+1:IFDD=0ANDNC>2THENDD=-1:NC=1
```

Set up the Octon array.

```
1200 Y=0:I=0:GOSUB1220:Y=5:GOSUB1220:Y=7:GOSUB1220:Y=8:GOSUB1220
1210 Y=4:I=8:GOSUB1220:Y=6:GOSUB1220:Y=3:I=-3:GOSUB1220:GOTO1230
1220 FORX=1TO8:C(X,Y)=I:NEXTX:RETURN
1230 FORI=1TO8:READX,Y:C(I,1)=X:C(I,2)=Y:NEXTI
1240 DATA 5,1,9,13,13,5,1,9,9,1,5,13,13,9,1,5
1250 FORI=1TONC:C(I,0)=-1:NEXTI:HT=0
```

Display the arena and place the Octons.

```
1260 CLS:FORX=26TO59:SET(X,2):SET(X,42):NEXT
1270 FORY=15TO29:SET(0,Y):SET(1,Y):SET(84,Y):SET(85,Y):NEXT
1280 Y=14:FORX=2TO24 STEP2:SET(X,Y):SET(X+1,Y):SET(84-X,Y):SET(8
5-X,Y):Y=Y-1:NEXT
1290 Y=30:FORX=2TO24 STEP2:SET(X,Y):SET(X+1,Y):SET(84-X,Y):SET(8
5-X,Y):Y=Y+1:NEXT
1300 READX,Y:IFX=0THEN1320ELSEFORI=XTOYSTEP3:PRINT@I," ";:NEXTI:
GOTO1300
1310 DATA 79,91,140,158,201,225,262,292,323,359,387,402,408,423,
451,463,475,487,515,530,536,551,579,615,646,676,713,737,780,798,
847,859,0,0
1320 FORX=41TO44:SET(X,19):SET(X,25):NEXT:FORX=39TO40:SET(X,20):
SET(X+6,20):SET(X,24):SET(X+6,24):NEXT:FORY=21TO23:SET(37,Y):SET
(38,Y):SET(47,Y):SET(48,Y):NEXT
```

```
1330 A$=CHR*(130)+CHR*(129):PRINT@269,A$;:PRINT@284,A$;:PRINT@39
4,A$;:PRINT@415,A$;:PRINT@586,A$;:PRINT@607,A$;:PRINT@717,A$;:PR
INT@732,A$;
1340 A$=CHR*(160)+CHR*(144):PRINT@205,A$;:PRINT@220,A$;:PRINT@33
0,A$;:PRINT@351,A$;:PRINT@522,A$;:PRINT@543,A$;:PRINT@653,A$;:PR
INT@668,A$;
1350 PRINT@834,"ARENA":PRINT@897,"OF OCTOS";
1360 PRINT@811,"H 1 2 3 4 5 6 7 8":PRINT@866,"SHIELD: ";:PRINT@
928,"STRENGTH: ";
1370 FORI=1TONC:PRINT@64*(I,2)+3*(I,1)-1,I;:PRINT@876+2*I,"N
";:PRINT@940+2*I,"B";:NEXTI
1380 PRINT@875,"N";:PRINT@938,"10";
```

Place the human at random about the firepit.

```
1390 HT=-1:PX=7:PY=4:P$="H":HS=11:GOSUB310:GOSUB270:HD=0:HM=10:H
L=0:IFDDTHEN3000ELSEPRINT@98,"PRACTICE":PRINT@164,"COMBAT":GOT
O3000
```

Alternate HT between 0 and -1.

```
2000 HT=-(HT+1):IFHTTHEN3000
```

Octon's turn. If Octon has been removed, go to the next one.

```
2010 FORI=1TONC:PRINT@468,F*(I);:IFC(I,0)=0THEN2620
```

Define the player string. Display it and clear the message display. Number of moves starts at 0: human "chance of hit" set to -3.

```
2020 P$=RIGHT$(STR$(I),1):PRINT@46,"OCTON NUMBER ";P$;:GOSUB260:
C(I,7)=0:HC=-3
```

If Octon is unshielded, show it.

```
2030 IFC(I,5)THENGOSUB210ELSEGOSUB220
```

Increment number of turns. If done, go to the next Octon.

```
2040 C(I,7)=C(I,7)+1:IFC(I,7)>C(I,6)THEN2620
```

Find out which Octon is the weakest.

```
2050 WK=-1:FORL=1TONC:PRINT@468,F*(L);:IFC(L,0)=0THEN2060ELSEIFC
(L,4)<C(I,4)THENWK=0:GOTO2060ELSE2060
2060 NEXTL
```

Define CX and CY. Get direction and distance to human.

```
2070 CX=C(I,1):CY=C(I,2):GOSUB30
```

CL is the number of turns left.

```
2080 CL=C(I,6)-C(I,7)+1
```

Octon's logic. Depending on factors such as distance, number of moves remaining, which Octon is weakest, and human strength, the Octon will either move, raise shield, or strike.

```
2090 IFCL>2THEN2250
2100 IFCL>1THEN2140
2110 IFDS<>1THEN2340
2120 IFHS=1ORLF>1THEN2130ELSE2340
2130 IFWKTHEN2340ELSE2350
2140 IFDS<>1THEN2180
2150 IFLF>1THEN2170
2160 IFHS<=2THEN2350ELSE2380
```

```

2170 IFWKTHEN2340ELSE2350
2180 IFDS>2THEN2220
2190 IFLF>1THEN2210
2200 IFHS<=1THEN2370ELSE2380
2210 IFWKTHEN2380ELSE2370
2220 IFLF>1THEN2240
2230 IFC(I,4)>HS-6THEN2370ELSE2380
2240 IFWKTHEN2380ELSE2370
2250 IFDS<>1THEN2310
2260 IFLF>1THEN2280
2270 IFSBTHENS=0:AN=-1:GOTO2380ELSE2350
2280 IFWKTHEN2300
2290 IFSBTHENS=0:GOTO2360ELSE2350
2300 IFSBTHENS=0:GOTO2380ELSE2350
2310 IFLF>1THEN2330
2320 GOTO2370
2330 IFWKTHEN2380ELSE2370

```

Octon raises shield.

```

2340 GOSUB210:M$="RAISE SHIELD":GOSUB230:GOTO2040
2350 SB=-1:GOSUB2580:GOTO2040
2360 GOSUB2460
2370 PX=C(I,1):PY=C(I,2):GOSUB120:GOSUB2390:GOTO2040
2380 IFANTHENAN=0:GOSUB2460:GOTO2370ELSEMV=MV-2*(MV-5):RT=-1:GOTO2370

```

Octon moves. Set message, depending on retreat flag.

```
2390 M$="ADVANCE":IFRTTHENM$="RETREAT":RT=0
```

Check for "out of arena."

```
2400 GOSUB280:IFANTHENM$=M$+" & HIT WALL":GOSUB230:RETURN
```

Check for "in pit."

```

2410 GOSUB290:IFF6THENM$=M$+" INTO PIT":GOSUB230:PRINT@64*PY+3*PX, ".":GOSUB310:IFC(I,4)<>0THENRETURNELSEPRINT@64*PY+3*PX, ".":C(I,7)=8:RETURN

```

Make sure that the Octon isn't moving into a space occupied by the human or another Octon.

```

2420 IFHX=NXANDHY=NYTHENGOSUB2460:GOTO2390
2430 FORJ=1TOUNC:PRINT@468,F$(J):IFC(J,0)=0THEN2450
2440 IFC(J,1)=NXANDC(J,2)=NYTHENGOSUB2460:GOTO2390
2450 NEXTJ:GOTO2560

```

If there is an obstruction, go around it.

```

2460 RN=RND(2):RN=RN-2:QNMV6GOTO2470,2480,2490,2500,,2510,2520,2530,2540
2470 IFRNTHENMV=2:GOTO2550ELSEMV=4:GOTO2550
2480 IFRNTHENMV=1:GOTO2550ELSEMV=3:GOTO2550
2490 IFRNTHENMV=2:GOTO2550ELSEMV=6:GOTO2550
2500 IFRNTHENMV=1:GOTO2550ELSEMV=7:GOTO2550
2510 IFRNTHENMV=3:GOTO2550ELSEMV=9:GOTO2550
2520 IFRNTHENMV=4:GOTO2550ELSEMV=8:GOTO2550
2530 IFRNTHENMV=7:GOTO2550ELSEMV=9:GOTO2550
2540 IFRNTHENMV=6:GOTO2550ELSEMV=8:GOTO2550
2550 GOSUB120:RETURN

```

If there are no obstructions, show the Octon and check for a stone.

```
2560 GOSUB270:C(I,1)=NX:C(I,2)=NY:GOSUB300:IFNOTF6THENGOSUB230:RETURN
```

Octon has passed a stone: There is a 50% chance of tripping and losing a turn.

```
2570 IFRND(2)=1THENGOSUB230:RETURNELSEM$=M$+" AND TRIP":GOSUB230:GOSUB220:C(I,7)=C(I,7)+1:IFC(I,7)>C(I,6)THENRETURNELSEM$="GETTING UP":GG=64*C(I,2)+3*C(I,1):PRINT@66,CHR$(95):GOSUB230:PRINT@66,P$:RETURN
```

Octon swings. Set message and lower shield. If human is unshielded, then hit; otherwise increase human's "chance of hit" variable.

```
2580 M$="SWING":GOSUB220:IFNOTHDTHEN2600ELSEHC=HC+4:IFHC>9THENHC=9
```

Find out if Octon has hit human.

```
2590 IFRND(10)>HCTHENM$=M$+" - DEFLECTED":GOSUB230:RETURN
```

Human hit; add to the message and print it. Decrement human strength, and show it.

```
2600 M$=M$+" AND HIT":GOSUB230:HS=HS-1:IFHS>0THENPRINT@938,HS:RETURN
```

The human has lost all strength points, and loses. Show the human removed, and go to the closing statements.

```
2610 PRINT@64*HY+3*HX, ".":PRINT@938,HS:FORN=1TOUNC:FORJ=1TOUNC:PRINT@468,F$(J):NEXTJ,N:GOTO6000
```

Go on to the next Octon's turn. If it's the last one, switch HT and go to the human's turn.

```
2620 FORN=1TOUNC:FORJ=1TOUNC:PRINT@468,F$(J):NEXTJ,N:GOSUB260:NEXTI:GOTO2000
```

Human's turn. Show heading, clear the message display, define player string, and set number of turns left to 0.

```
3000 PRINT@46,"HUMAN'S TURN ";:GOSUB260:P$="H":HL=0
```

If human is defended, show it.

```
3010 IFHDTHENGOSUB210ELSEGOSUB220
```

Set all Octon "chance of hit" variables to -3.

```
3020 FORI=1TOUNC:PRINT@468,F$(I):C(I,3)=-3:NEXTI
```

Increment number of turns; if done, switch HT and let Octons take over.

```
3030 HL=HL+1:IFHL>HMTHEN2000
```

Input routine. (This also keeps the fire going.)

```

3040 I=26
3050 I=I-1:IFI=0THEN3040
3060 A$=INKEY$:PRINT@468,F$(I):IFA$=""THEN3050

```

continued on next page

continued from previous page

If A\$ is a number, or one of the alternate movement keys, go to the move subroutine.

```
3070 IFA$="1"ANDA$<="9"THEN3190
3075 IFA$="U"THENA$="7":GOTO3190 ELSEIFA$="I"THENA$="8":GOTO3190
ELSEIFA$="O"THENA$="9":GOTO3190 ELSEIFA$="J"THENA$="4":GOTO3190
3076 IFA$="L"THENA$="6":GOTO3190 ELSEIFA$="M"THENA$="1":GOTO3190
ELSEIFA$=","THENA$="2":GOTO3190 ELSEIFA$="."THENA$="3":GOTO3190
```

If A\$ is not one of the other three valid keys, return.

```
3080 IFA$<>"S"ANDA$<>"R"ANDA$<>"P"THEN3050
```

The human raises his shield.

```
3090 IFA$="R"THENM$="RAISE SHIELD":GOSUB230:GOSUB210:GOTO3030
```

The human passes.

```
3100 IFA$="P"THENM$="PASS":GOSUB230:FORN=1TO4:FORJ=1TO25:PRINT@468,F$(J);:NEXTJ,N:GOTO2000
```

The human swings. Check to see which Octon is next to him.

```
3110 M$="SWING":GOSUB220:FORI=1TONC:PRINT@468,F$(I+B);:IFC(I,0)=0THEN3140
```

K and L will be 1 if Octon is nearby.

```
3120 K=ABS(C(I,1)-HX):L=ABS(C(I,2)-HY)
```

Set Octon's "I'm hit" variable if so.

```
3130 IFK<=1ANDL<=1THENC(I,8)--1ELSEC(I,8)=0
```

Look again to see if any Octons were hit.

```
3140 NEXTI:FORI=1TONC:PRINT@468,F$(I);:IFC(I,0)=0ORC(I,8)=0THEN3180
```

Increase Octon's chance of getting hit.

```
3150 C(I,3)=C(I,3)+4:IFC(I,3)>9THENC(I,3)=9
```

Octon will be hit if undefended, or if the random number comes up. If he loses all his strength points, remove him.

```
3160 IFC(I,5)=0ORRND(10)<C(I,3)THENM$=M$+" AND HIT":GOSUB230:C(I,4)=C(I,4)-1:HT=0:GOSUB250:HT=-1:GOSUB90:IFFGTHENC(I,0)=0:GOSUB100:PRINT@64*(I,2)+3*(I,1),".":GOTO3030:ELSE3030
```

The strike is deflected; show it and return.

```
3170 M$=M$+" - DEFLECTED":GOSUB230:GOTO3030
```

The human has swung his sword and there was no one there.

```
3180 NEXTI:M$=M$+" AND MISSED":GOSUB230:GOTO3030
```

The human moves. If A\$ was a 5, go back and try again.

```
3190 IFA$="5"THEN3040ELSEM$="MOVE"
```

Define the variable MV. Go set the new X and Y positions, and check for "out of arena."

```
3200 MV=VAL(A$):PX=HX:PY=HY:GOSUB120:GOSUB280
3210 IFANTHENM$=M$+" AND HIT WALL":GOSUB230:GOTO3030
```

Loop to see if there is an Octon in the way. If so, you are pushing him; if not, next Octon.

```
3220 FORI=1TONC:PRINT@468,F$(I);:IFC(I,0)=0THEN3240
3230 IFC(I,1)=NXANDC(I,2)=NYTHEN3280
3240 NEXTI
```

Check for firepit; if yes, go to the pit subroutine.

```
3250 IFBD(NX,NY)--1THENM$=M$+" INTO PIT!":GOSUB230:PRINT@64*PY+3*PX, ".":GOSUB310:GOTO3030
```

Place the human and check to see if there was a stone. If not, or if so and luck is in your favor, then no problem. Otherwise lose the turn.

```
3260 GOSUB270:HX=NX:HY=NY:GOSUB300:IFNOTFBOR(FGANDRND(2)>1)THENGOSUB230:GOTO3030
```

```
3270 M$=M$+" AND TRIPS":GOSUB220:GOSUB230:HL=HL+1:IFHL>HMTHEN2000ELSEGG=64*HY+3*HX:PRINT@66,CHR$(95);:M$="GETTING UP":GOSUB230:FORK=1TO3:FORM=1TO25:PRINT@468,F$(M);:NEXTM,K:PRINT@66,"H";:GOTO3030
```

Human pushes Octon. Save human's present location, then set the location to which the Octon is being pushed.

```
3280 X1=NX:Y1=NY:PX=NX:PY=NY:GOSUB120
```

Define the message and check to see if the Octon's pushed position is out of the arena. If so, Octon's chance of hit increases.

```
3290 M$="PUSH":GOSUB280:IFANTHENC(I,3)=C(I,3)+4:IFC(I,3)>9THENC(I,3)=9
```

Also, show the message and go to the next turn.

```
3300 IFANTHENM$=M$+" OCTON TO WALL":GOSUB230:GOTO3030
```

Check to see if Octon's pushed position is already occupied.

```
3310 FORJ=1TONC:PRINT@468,F$(J);:IFC(J,0)=0THEN3330
3320 IFC(J,1)=NXANDC(J,2)=NYTHENC(I,3)=C(I,3)+4:GOSUB230:IFC(I,3)>9THENC(I,3)=9:GOTO3030ELSE3030
```

Move the human to his new position.

```
3330 NEXTJ:PRINT@64*HY+3*HX, ".":PRINT@64*Y1+3*X1,"H";:HX=X1:HY=Y1
```

Check to see if Octon has been pushed into firepit; place him in his position.

```
3340 IFBD(NX,NY)--1THENM$=M$+" INTO FIREPIT":GOSUB230:HT=0:P$=RIGHT$(STR$(I),1):GOSUB310:PRINT@64*Y1+3*X1,"H";:HT=-1:P$="H":IFHDTHENGOSUB210:GOTO3030ELSEGOTO3030
```

```
3350 PRINT@64*NY+3*NX,RIGHT$(STR$(I),1);:C(I,1)=NX:C(I,2)=NY:GOSUB300
```

Check to see if Octon has been pushed across stone.

```
3360 IFFGTHENM$=M$+" ACROSS STONE":GOSUB230:GOTO3030ELSEGOSUB230:GOTO3030
```

Human has won.

```

4000 G$="":IFNC>1THENG$="S"
4010 CLS:PRINT@325,"CONGRATULATIONS! YOU HAVE SUCCEEDED IN FIGHT
ING"NC"OCTON
WARRIOR"G$".":PRINT
4020 IFDDTHEN4080ELSEIFNC=2THEN4060
4030 PRINT"
YOU HAVE COMPLETED THE FIRST HALF OF TRAINING."
4040 PRINT"
TO CONTINUE TRAINING, PRESS <ENTER>";
4050 INPUTA:RESTORE:GOTO1190
4060 PRINT"
THIS IS THE END OF YOUR TRAINING. YOU ARE NOW READY TO
START THE CONTEST."
4070 PRINT"
PRESS <ENTER> TO BEGIN";:INPUTA:RESTORE:GOTO1190
4080 IFNC=8THEN4110ELSEG$="ST":IFNC=2THENG$="ND"ELSEIFNC=3THENG$
="RD"ELSEIFNC>3THENG$="TH"
4090 PRINT"YOU HAVE COMPLETED THE"STR$(NC);G$" CONTEST IN THE OC
TON CHALLENGE.
YOU MAY REST IF YOU WISH, THEN PRESS ANY KEY TO CONTINUE."
4100 A$=INKEY$:IFA$=""THEN4100ELSENC=NC+1:RESTORE:GOTO1200
4110 CLS:PRINTCHR$(23):PRINT@320,"YOU HAVE WON THE OCTON CHALLENGE"
4120 PRINT"
BUT REMEMBER... IF YOU SHOW ANY
SIGN OF COWARDICE (EVEN IF YOU
TYPE 'RUN') YOU WILL BE
CAPTURED ONCE AGAIN...":PRINT:END
4130 GOTO4130

```

Instructions.

```

5000 CLS:PRINT@20,"ARENA OF OCTOS"
5010 PRINT"

```

YOUR STAR CRUISER HAS BEEN CAPTURED WHILE TRAVELING THROUGH SPACE CLAIMED BY THE OCTON DYNASTY. AFTER SOME DELIBERATION, THE COUNCIL OF EIGHT HAS GRANTED YOU THE CHANCE TO EARN YOUR FREEDOM BY ACCEPTING THE 'OCTON CHALLENGE'.

YOU WILL BE PLACED IN AN ARENA, WITH A SWORD AND A SHIELD. YOU WILL THEN ENGAGE ONE OCTON IN COMBAT. IF YOU SURVIVE, YOU WILL THEN FIGHT TWO, THEN THREE.... UNTIL THERE ARE EIGHT OCTONS IN THE ARENA.

THE OCTONS HAVE EIGHT MOVES, AND A 'STRENGTH' FACTOR OF EIGHT. THIS MEANS THAT AN OCTON CAN RECEIVE EIGHT DIRECT BLOWS BEFORE HE IS REMOVED FROM THE ARENA.

```

5040 PRINT@976,"PRESS <ENTER> TO CONTINUE";
5050 A$=INKEY$:IFA$=""THEN5050
5060 CLS:PRINT"

```

THE ARENA OF OCTOS IS AN EIGHT-SIDED STRUCTURE. THERE ARE EIGHT STONES PLACED IN THE ARENA, TO TRIP THE CONTESTANTS. IN THE CENTER OF THE ARENA IS ANOTHER HAZARD, THE FIRE PIT.

THE COUNCIL OF EIGHT HAS DETERMINED THAT HUMANS ARE TO BE ALLOWED TEN MOVES, AND TEN STRENGTH POINTS SINCE HUMANS ARE DECIMAL CREATURES.

IF YOU CROSS A STONE DURING THE CONTEST, YOU HAVE A 50% CHANCE OF TRIPPING AND LOSING A TURN. IF YOU ENTER THE FIREPIT, YOU WILL LOSE ONE STRENGTH POINT, AND YOU WILL LOWER YOUR SHIELD AND BLINDLY ESCAPE IN ANY DIRECTION.

```

5090 PRINT@976,"PRESS <ENTER> TO CONTINUE";
5100 A$=INKEY$:IFA$=""THEN5100
5110 CLS:PRINT@47,"7";:PRINT@55,"8";:PRINT@63,"9";:PRINT@175,"4"
;:PRINT@191,"6";:PRINT@303,"1";:PRINT@311,"2";:PRINT@319,"3";
5120 FORX=98TO123:SET(X,7):NEXT:FORY=3TO10:SET(110,Y):SET(111,Y)
:NEXT:Y=2:FORX=98TO106STEP2:SET(X,Y):SET(X+1,Y):SET(X+16,Y+6):SE
T(X+17,Y+6):Y=Y+1:NEXT

```

```

5130 Y=12:FORX=98TO106STEP2:SET(X,Y):SET(X+1,Y):SET(X+16,Y-6):SE
T(X+17,Y-6):Y=Y-1:NEXT
5140 PRINT@0," YOU MOVE IN THE ARENA BY";:PRINT@64,"PRESSJ
NG KEYS 1-9 OR THE LETTER KEYS";:PRINT@128,"INDICATED BY THE CHA
RT, TO MOVE IN";
5150 PRINT@192,"ONE OF THE EIGHT DIFFERENT";:PRINT@256,"DIRECTIO
NS.";
5160 PRINT@384," YOUR ONLY OTHER COMMANDS ARE:
<S> SWING YOUR SWORD
<R> RAISE YOUR SHIELD
<P> PASS (AND FORFEIT REMAINING MOVES)"
5170 PRINT"

```

IF AN OCTON HAS HIS SHIELD RAISED, YOUR FIRST SWING HAS A 10% CHANCE OF CONNECTING. THE SECOND SWING HAS A 50% CHANCE, AND AFTER THAT THERE IS A 90% CHANCE OF A HIT.

```

5180 PRINT@976,"PRESS <ENTER> TO CONTINUE";
5190 A$=INKEY$:IFA$=""THEN5190
5200 CLS:PRINT@256,"

```

THE COUNCIL OF EIGHT HAS GRANTED YOU A 'PRACTICE' SESSION, TO FAMILIARIZE YOU WITH THE ARENA.

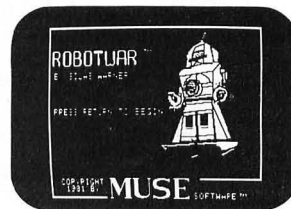
Human has lost.

```

6000 G$="":IFNC>1THENG$="S"
6010 CLS:PRINT@325,"YOU HAVE LOST IN YOUR BATTLE AGAINST"NC"OCTO
N WARRIOR"G$".":IFDDTHENPRINT"
YOU WILL NOW REMAIN AN OCTON SLAVE, AND BE FORCED TO WORK AT
THE OCTAL PALACE FOR SCRAPS OF FOOD.":PRINT"
YOU WILL HAVE TO TRY AGAIN WHEN YOU THINK YOU ARE "E
ND
6020 PRINT"
SINCE THIS IS A PRACTICE SESSION, YOU ARE ALLOWED TO TRY AGAIN."
6030 PRINT"
PRESS <ENTER> TO CONTINUE PRACTICE";:INPUTA:RESTORE:GOTO1200

```

WHAT MAKES ROBOTWAR™ UNIQUE?



YOU.

RobotWar™ is *not* a manual dexterity game. You write a special Battle Program for your own robots in a true battle of wits with the demo robots or robots designed by other combatants. You program the robot's

- velocity
- laser cannon aim
- radar direction
- battlefield positioning

You are responsible for your robot's success or ultimate destruction. The basics of Battle Language are easily learned. Yet, like chess, RobotWar™ may take a lifetime to master.

MUSE SOFTWARE™

Apple II is a trademark of Apple Computer Corp.

330 N. CHARLES STREET
BALTIMORE, MD 21201
(301) 659-7212

Call or write for information and the name of your nearest MUSE dealer

Now for the TRS-80™ Models I & III

LDOS™

THE TRS-80™ OPERATING SYSTEM FEATURING:

- * **DOUBLE** Sides & **DOUBLE** Density support.
- * **AUTOMATIC** Density recognition.
- * 35, 40, 77, 80 and any other track counts are supported.
- * All available drive stepping rates are supported.
- * Hard Drive support, can be **HANDLED AS A SINGLE DRIVE**.
- * Hard drive partitioning, one hard drive can act as up to six drives.
- * Intermix 5", 8" and Hard drives, up to a total of 8 drives.
- Compatible with the Model I Radio Shack Expansion Interface.
- Upward compatible with **TRSDOS** (2.3 & 1.3 as documented).
- Fully supports Microsoft language products, without modification.
- Complete media compatibility Model I to Model III and Model III to Model I.
- Full support for **LOBO's LX-80** interface.
- Full support for **PERCOM's DOUBLER II**.
- Complete documentation (well over 250 pages).
- Complete technical information.
- A **TOLL-FREE** 800 number for customer service.
- An **LDOS** users bulletin board on MicroNET.
- A Quarterly **LDOS** users magazine (The **LDOS QUARTERLY**).
- A liberal update policy.
- An enhanced **BASIC (LBASIC)** including:
 - > Upward compatible with Microsoft Basic.
 - > High speed **LOAD** and **SAVE**.
 - > Run multiple programs with common variables.
 - > **BLOCKED** (variable length) files are supported.
 - > **DOS** commands may be executed from **LBASIC**.
 - > Built in string array **SORT**.

- > Single stepper for debugging.
- > Several new statements and file modes.
- A compiled **JOB CONTROL LANGUAGE (JCL)**.
- An Extended Debugging and Monitor program (with disk access).
- **CMDFILE** for movement of disk and/or tape system (**/CMD**) files.
- Device independent operation.
- Full **LINKing**, **ROUTEing**, **FILTERing** and **SETting** are supported.
- **MiniDOS** feature for constant access to certain **DOS** commands.
- Sophisticated communications software included.
- Wildcard characters and partial Filespecs are supported.
- Each file is **DATED**, showing when it was last modified.
- Backup: Mirror, by Class, if Modified, by Date, by Extension, etc.
- Selectable **PURGE** for fast disk "cleanup" of unwanted files.
- Print formatter, for control of printer output.
- Built in printer **SPOOLER**, to both disk and memory.
- Joblog to record all system operations with time stamps.
- **UPPER** and **lower** case support, throughout the system.
- Blinking cursor with selectable cursor character.
- 128 character **TYPE AHEAD** buffering for keyboard input.
- Assign strings to individual keys with Key Stroke Multiply (**KSM**).
- **SUPER FAST** operation with the **SYSRES** feature.
- Extensive user control and system feedback.
- Complete transportability of software among all **Z-80 LDOS** systems through the use of the **LDOS** high memory supervisory call system.
- Dealers to serve users Nationwide and in the Common Market.
- The only **DOS** with a limited **ONE YEAR WARRANTY!**
- Enjoy a professional operating system on **YOUR TRS-80!**

* Specific hardware is required to use these features.

The Ultimate In
Operating Systems
For Model I & III

Only \$**169**⁰⁰
Version 5.1

Model I #25-265001D
Model III #27-265001D

* Model I LDOS provided on 35 track single density media.
* Model III LDOS provided on 40 track double density media.
* Prices & Specifications are subject to change without notice.

**LOGICAL
SYSTEMS
INC.**



TSE-HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

**DEMAND THE MOST
FOR YOUR
WORD PROCESSING
DOLLAR?
THEN MOVE UP
TO PROSOFT'S
NewScript . . .**

- * Easy-to-use Full Screen Editing
- * Typeahead — never loses keystrokes
- * Headings, Page Numbering, Centering
- * Form letters, Big Documents
- * Global search and change
- * Double-Width, Underlining
- * Sub-scripts, Super-scripts
- * Table of Contents, Indexing
- * Proportional font right-margin justification on 737, 739, L.P. IV, D.W. II
- * All 12 MX-80 fonts + underlining
- * Italics on GRAFTRAX MX-80
- * Supports Diablo, Microline, Anadex
- * 160 pages of excellent documentation includes "EZEDIT", "EZSCRIPT", self-study tutorial, and hundreds of examples
- * Runs under TRSDOS, NEWDOS, NEWDOS/80, LDOS, DOSPLUS.
- * Requires 48K TRS-80 with one disk drive.
- * Specify: Model I or Model III.

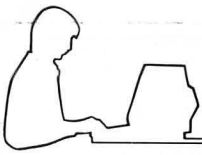
NewScript (Mod-I) #25-269001D \$99.95
Mailing Labels Option #25-269002D \$14.95

NewScript (Mod-III) #27-269001D \$99.95
Mailing Labels Option #27-269002D \$14.95



TSE-I-HARDSIDE
 6 South St., Milford, NH 03055 (603) 673-5144
 TOLL FREE OUT-OF-STATE 1-800-258-1790

PROSOFT



CONTEST RESULTS

The SoftSide "BASICally Speaking" Programming Contest

In the April issue of **SoftSide**, we announced "The Most Unusual Programming Contest You've Ever Heard Of." The point of the contest was to write one or more lines of proper BASIC code which also made sense as English. Several examples were given, including the stimulating one-liner,

IF YOU = GOOD THEN YOU = WINNER

The race was on to submit the most creative BASIC/English code in **SoftSide** history!

After several weeks of fretful anticipation, we received our first contest entry. Close upon the heels of that one (a few weeks later) we received yet another. A third poured in only a few days after that. On one memorable day, two entries arrived--one of them from Holland, which proved to be almost too much excitement for one of our editors of Dutch ancestry. In the end, almost ten people (well, seven, to be exact) deluged us with their unusual entries, the last arriving on the very eve of the contest's official closing.

Needless to say, we were swamped with work, processing all the entries. That's why this report on the results was delayed until October.

But here, at last, for all those who have been waiting with bated breath (all seven of you) for the announcement of the winners, is the final decision of the judges. Having received no worthwhile offers of bribes along with the contest entries, we hereby declare that nobody won first prize. (Too bad — that "Round the World in 80 Days" luxury cruise for two would have given us some good publicity.) However, we would like to give credit to all those who sent entries, and we feel that three of them deserve honorable mention.

The "Most Philosophical" award goes to Terry David Clark of Columbus, Indiana. Terry's entry is written for S-80 Disk BASIC:

```
10 REMEMBER TO MAINTAIN A
20 CLEAR &HEAD
30 REMEMBER ALSO THAT WE
HAVE A PRECIOUS RIGHT TO
40 PRINT "THE WORD":
REMEMBER WHEN YOU WERE
50 AWAKENED = &HAD:
REMEMBERED WHAT THE WISE
MEN TELL US.
60 WISDOM = AWAKEN:
BRAIN = WISDOM: SURVIVE =
```

```
WISDOM: YOU = BRAIN
70 REMEMBER THOUGH THAT
WE MUST ALSO CONTINUE THE
QUEST
80 FOR QUESTING = THEWILL
TO SURVIVE
90 IF WISDOM > STRENGTH
THEN PRINT "IS GREATER
THAN THE SWORD."
100 REMEMBER ALSO TO ASK
THE
110 NEXT QUEST'ION
120 REMEMBER IT? HOW CAN I
SURVIVE?
130 IF (SURVIVE = WISDOM) AND
(WISDOM = BRAINS) AND
(BRAIN = YOU) THEN READ
BOOK$
140 FOR YOU = THEWILL TO
SURVIVE + BRAINS
150 IF YOU = >WISDOM THEN
PRINT BOOK$
160 NEXT YOU: REMEMBER THE
PAST, BECAUSE THE
170 DATA "WE LEARN FROM
THE PAST AND APPLY TO THE
PRESENT"
180 REMEMBER! CAN SAVE US
FROM THE
190 END
```

The award for the "Best One-Liner" goes to Ron Kohavi and Moshe Nissim of New York City, for this graphic Applesoft entry:

```
10 IF YOU = ARTIST THEN
DRAW ME AT LAST, PLEASE
```

And finally, the "Most Romantic" award goes to David Whitney of Oklahoma City, Oklahoma, for this S-80 contribution:

```
10 MYHEARTSTRUEFEELINGS =
"I LOVE YOU"
20 PRINT
MYHEARTSTRUEFEELINGS
OVER AND OVERAGAIN
30 FOR I = HAVE TO
TELLYOUILOVEYOU
40 ILOVE = (YOU AND NOOTHER)
50 IF U = LOVEME THEN
MARRY = ME ELSE ILLBE = SAD
60 FOR MARRIAGE = NOW TO
DEATHDOUSPART
```

We enjoyed and appreciated the other entries as well, which were submitted by the following people:
Elrhea Bigham, Van Wert, Ohio
David Horwat, Morris Plains, New Jersey
Bill Katz, Tokyo, Japan
Huub Sanders, the Netherlands

K-Byters

ANOTHER PROGRAMMING CHALLENGE

Last summer **SoftSide** began inviting its readers to submit "One Liners" — self-contained, single-line programs for the S-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 is not an absolute limit.

Here, then, are the official rules:

1. The program must be written for the Apple, S-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





MISSILE COMMAND

An Applesoft K-Byter by Mike Watson, Beaverton, OR

In this HIRES graphics game the object is to destroy as many of the incoming ICBMs as you can. You have three bases from which to fire, each with a stockpile of ten ABMs. A base which is hit directly by an incoming missile is destroyed. Each ICBM that you destroy adds 50 points to your score; each one that gets past you deducts 100. Use the game paddles to position the crosshairs on the target, and press a number key (1, 2, or 3) to fire from the appropriate base.

```

3 TEXT
5 HOME : VTAB 12: PRINT TAB( 13
) "MISSILE COMMAND": GOSUB 500
0: DEF FN O(X) = VAL ( CHR#
(X - 128)): V = 21: A = 5: B =
5
6 HGR : HCOLOR= 1: FOR I = 1 TO
3: B(I) = 10: HPLLOT 0, I + 156
TO 279, I + 156: NEXT : SCALE=
2: ROT= 0: HCOLOR= 3: HPLLOT
0, 156 TO 8, 150 TO 16, 156: HPLLOT
132, 156 TO 140, 150 TO 148, 15
6: HPLLOT 279, 156 TO 271, 150 TO
263, 156

```

```

7 FOR I = 1 TO 3: READ BX(I), T(I
): NEXT
110 GOSUB 200: GOSUB 540
130 M = B(1) + B(2) + B(3): C = C +
INT ( RND (1) * 3) - 1: IF
NOT M THEN TEXT : HOME : HTAB
17: FLASH : PRINT "THE END":
NORMAL : END
140 GOTO 110
200 X = PEEK ( - 16384): POKE -
16368, 0: IF X > 176 THEN IF
B( FN O(X)) AND FN O(X) < 4
THEN 420
210 HCOLOR= 0: GOSUB 610: A = ( PDL
(0) / 255) * 273 + 3: B = ( PDL
(1) / 255) * 156 + 3: HCOLOR=
3: GOSUB 610: RETURN
420 HPLLOT BX( FN O(X)), 150 TO A,
B: HCOLOR= 0: HPLLOT BX( FN O
(X)), 150 TO A, B: GOSUB 730: VTAB
V: HTAB T( FN O(X)): PRINT B
( FN O(X)): RETURN
540 IF Q = 156 THEN GOSUB 630
560 Q = Q + 1: HPLLOT C, Q: RETURN
570 GOSUB 680
580 E = E + 50: GOSUB 6000: VTAB
1: XDRAW 1 AT A, B: PRINT "":

```

```

DRAW 1 AT A, B: RETURN
610 HPLLOT A - 3, B TO A + 3, B: HPLLOT
A, B + 3 TO A, B - 3: RETURN
630 IF Q = 156 THEN E = E - 100:
HCOLOR= 3: DRAW 1 AT C, Q: GOSUB
6000
650 FOR I = 1 TO 3: IF C > BX(I)
- 8 AND C < BX(I) + 8 THEN
B(I) = 0
660 NEXT
680 C = INT ( RND (1) * 260) + 8
: Q = 0: RETURN
730 B( FN O(X)) = B( FN O(X)) - 1
: IF ABS (A - C) < 2 AND ABS
(B - Q) < 2 THEN 570
735 XDRAW 1 AT A, B: FOR I = 1 TO
50: NEXT : DRAW 1 AT A, B: RETURN
5000 FOR I = 21000 TO 21037: READ
A: POKE I, A: NEXT : DATA
1, 0, 4, 0, 73, 60, 56, 56, 30, 247,
54, 46, 46, 53, 101, 12, 13, 24, 31,
15, 24, 247, 11, 24, 12, 24, 136, 15
8, 18, 101, 184, 227, 107, 8, 15, 24
, 4, 0, 8, 2, 140, 20, 271, 38
5010 POKE 232, B: POKE 233, 82: GOTO
680
6000 VTAB 22: PRINT "SCORE:" E: RETURN

```

STELLAR INCURSION

An S-80 K-Byter by Ken Huffman, Westerville, OH

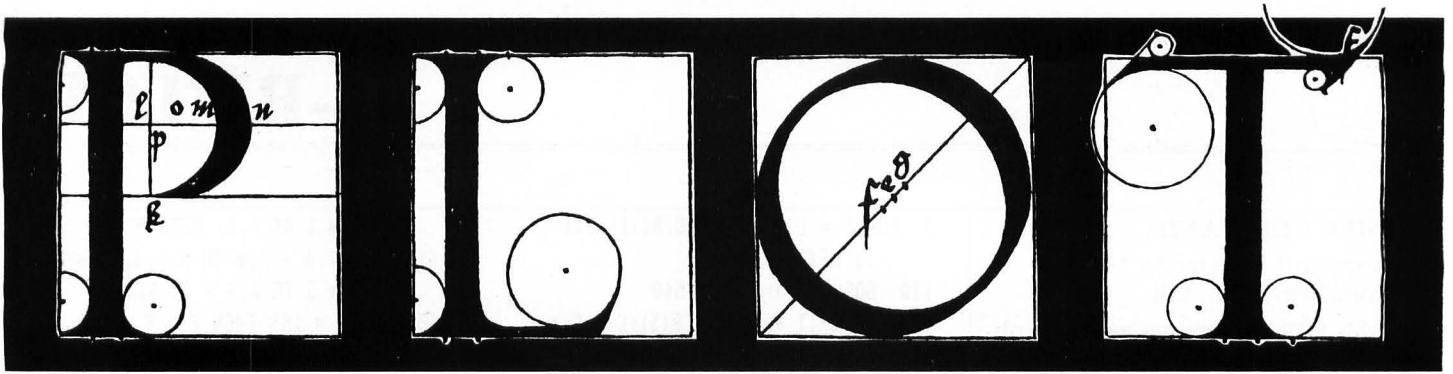
Your mission is to rid the galaxy of the menacing Cachexias, whose ships are shaped like X's. Your vessel, the infamous U.S.S. Antaeus, is the only mobile ship in the area. The lowly enemy has no defense against your phaser blasts, due to the shoddy construction of their ships. The vile creatures do not have the technology to repair their warships, so they haphazardly produce additional contraptions.

Your warship can carry only twenty phasers, but you can replenish them when you dock at a cross-shaped starbase. To do this you move your ship next to the starbase and point your phaser tube at it without firing. To fire your phasers, use the space bar. Shooting at stars is not recommended because phasers will deflect. Use the arrows to maneuver your ship, and the shift keys to rotate the phaser tube. You may select any of ten levels of play, to challenge the most proficient player.

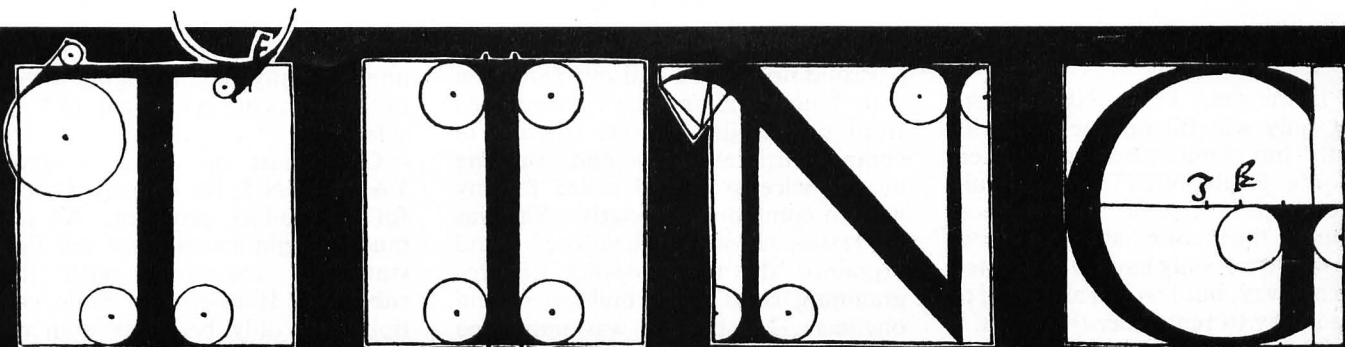
```

5 CLS: CLEAR 200: DEFINT D-Z: DEFSTR A-C: INPUT "1=HARD - 10=EASY": Q: DIM
P(99), A(23): A="!V!EEEBEQQB!E!REVEVRRVVRVQVVRVRRVVRVVBVVVRVVRV
VUVUVUVVRVVRVVRVVRVVRV": B=" . ": FORX=1 TO 69: B=B+CHR$(ASC(MID$(A,X)
)*3-67): NEXT: T=RND(8)-1
10 CLS: D=14400: E=140: K=15361: FORX=0 TO 23: A(X)=MID$(B, X*3+1, 3): NEX
T: FORX=0 TO 7: READ R(X), S(X): NEXT: DATA 2, 3, 4, -61, 1, -64, 3, -67, 2, -3, 4,
61, 1, 64, 3, 67: FORX=0 TO 14: FORY=0 TO 20: IFRND(15)=1H=5ELSE IFRND(E) < B+
125H=0ELSE H=6: G=6+1
15 PRINTA(H): NEXT: PRINT: NEXT: Q=Q*2: FORX=1 TO 4
20 M=RND(956): IFPEEK(M+K)=46PRINT@M, A(7): NEXTELSE 20
25 L=20: PRINT@984, "PHASERS LEFT:" L:
30 PRINT@M, A(T+SGN(L)*8+8): IFRND(Q)=1N=RND(956): IFPEEK(M+K)=46P
RINT@M, A(6): G=6+1
35 IFPEEK(D) < 1280RL=0GOTO70ELSE L=L-1: PRINT@997, L: P=M: S=T: J=0
40 P=P+S(S): P(J)=P: H=PEEK(P+K-1): I=PEEK(P+K): IFI=46PRINT@P, A(R(S
)): J=J+1: GOTO 40
45 IFI=EANDH=32S=(S+4+RND(7))AND 7: GOTO 40
50 IFI=EANDH=179J=J+1: G=6-1: IFG=0C="YOU HAVE SUCCEEDED.": Z=7
55 IFP=M: C="YOU NO LONGER EXIST.": Z=7: J=J+1
60 IFH=EANDI=191C="YOU HIT YOUR BASE.": Z=7: J=J+1
65 IFJFORH=0 TO J-1: PRINT@P(H), A(0): NEXT: IFZCLS: PRINT@470, C: END
70 R=PEEK(D): I=M: IFRAND8I=I-64
75 IFRAND16I=I+64
80 IFRAND32I=I-3
85 IFRAND64I=I+3
90 IFPEEK(I+K)=46PRINT@M, A(0): M=I
95 IFPEEK(D+64)T=(T+1)AND 7
100 IFPEEK(K+M+S(T))=19160T025ELSE 30

```



Within the following is a true history, and not an allegorical fantasy designed to convey a teaching moral, though it might well have been allegory. The lesson, if any, is left for the reader to discover. Please try the various versions of the program for enjoyment, for enlightenment, or for study as an interesting case history from the abnormal psychology of the bizarre — I had written an I Ching program for my Apple II, and was casting about for an interesting “hello” program as an introduction when I hit upon the idea that plotting the yang/yin symbol would be a perfect initiation to the Chinese symbol system. Fine, I thought, this will be simple: just a couple of circles with one being half the diameter of the other and split down the middle by two different centers. To plot such a circle one needs an x coordinate derived from the COS of a radian and a y coordinate derived from the SIN of that same radian. (Cont.)



THE YANG/YIN SYMBOL

by David Delli Quadri

The history of a search for execution speed in a simple H PLOT routine

Since a radian is a linear perimeter constant that represents the length of a degree's arc ($2 * \pi / 360$), a full circle would have $2 * \pi$ radians. A FOR...NEXT loop that runs from 0 to $2 * \pi$ will feed the required radians nicely, particularly if it is STEPIng at a rate of $2 * \pi / 200$, giving two hundred points to plot on the desired perimeter.

I decided to use the x axis at the 140 HGR line, the y at the 80, in order to center my figure. I also decided to vary the centers of my various circles along the $x = 140$ axis, moving from the $y = 80$ in the plotting of the smaller circles in order to maintain the figure in what I conceived to be an upright position. This necessitated starting the FOR...NEXT loop at $.5 * \pi$ rather than 0, so that the first x figured would be 0 plus the 140 baseline.

The Apple high resolution graphics screen, admirable in most respects, generates all those pretty moire figures because of the length discrepancy between the distances covered along the x and y axes. If one wishes to compensate for this enough to draw true circles rather than ellipses, a finagle factor of 1.2 must be multiplied into the x variable before it is added to the 140 baseline.

I put it all together to create the program listed in Table 1 as YANG/YIN-1. I ran it a few times just to make sure that it performed in the desired fashion (see Figure 1 for the screen output), then called my wife to come see so that I could show it off and bask in the glory of another successful program.

"That's nice Dear," my beloved said to me, "but doesn't the Yang/Yin symbol represent yang, the light, and yin, the dark, fading into one another as they circle, forming the Tao of Creation? I don't see any light in that figure. Besides, I would get so bored waiting for it to plot that I wouldn't sit through it to use your I-Ching."

As usual, she had a point. It was pretty slow at that. "Well, why not time it?" I thought. I loaded my trusty timing routine (see the first program in Table 1) which I wrote for the Mountain Hardware Apple Clock. I inserted the program to be timed in the middle of the routine by using the renumber/merge program Apple so thoughtfully provided, ran it, and lo, a 51 second execution time.

My wife's unwitting comment provided the stimulus for a new approach to the problem. If one plots from the inner set of perimeters to the (continued next page.)

outer in the same FOR...NEXT loop, it not only will fill up one side with "light," but should also prove faster, since the FOR...NEXT loop would only have to be passed through once. Of course, the second half of the outer circle would not only have to be plotted in the old way, but I would also have to figure a way to remember the last x, y point from which to plot. No problem. The good old Apple is great at remembering stuff. By re-arranging a few lines we now have YANG/YIN-2 in Table 1.

It performed as expected, cutting a full 13 seconds off the execution time, but somehow I could hear my wife mumbling about how slow it seemed. Could it be cleaned up to run more quickly? Just for starters, all those places in which computations were made each time a comparison was made could have the computation made once outside the loop, rather than 200 times in the loop. That should save some time! In Appendix E, "Speeding up your program," in the Applesoft Basic Programming Reference Manual (page 120), I found several other helpful hints: using variables instead of constants; putting the most used variables first; and figuring the time overhead involved in moving from one line to another.

As I attempted to put all these precepts for speedy execution together, I began to see that I could make a "one-liner" of my Yang/Yin plotter. That is, a program complete in one line that has many statements strung together. The problem to be overcome was fairly simple: Get around the blind trailing of unexecuted IF...THEN statements. The solution is, of course, to eliminate the IF...THEN statements. In analyzing the plotting in YANG/YIN-2, one finds the following to be true:

```
T < 1.5 * pi  H PLOT 140 + X, 80 + Y
TO 140 + X / 2, 110 + Y / 2
T > 1.5 * pi  H PLOT 140 + X/2, 50 +
Y/2 TO 140 + X/2, 50 - Y/2
```

By exploiting the little used and all but unknown Applesoft ability to evaluate and execute conditional multipliers, one can build a plotting statement that will emulate our multiple IF...THEN statements. A conditional inequality is ($T < 1.5 * \pi$) which will evaluate as 0 if the condition is false, and 1 if the condition is true. The statement $140 + X / (1 + 1 * (T > 1.5 * \pi))$ will be $140 + X$ when $T < 1.5 * \pi$, and $140 + X / 2$ when $T > 1.5 * \pi$. By making the necessary changes, YANG/YIN-3

was produced.

Proud now that I had my Yang/Yin with both light and dark, condensed from twelve lines to one (slightly in comprehensible) line, and running nearly twice as fast, I called for my wife to come be appreciative. She was impressed as I ran it, then listed it and explained the true elegance of programming conditional multipliers and oneliners. Mostly she was impressed that I felt compelled to continually test to find where the FOR...NEXT loop was in order to know what to plot.

"All that testing and multiplying! Doesn't the computer know that during the first half of the loop, T is always less than $1.5 * \pi$, and during the second half, it is always more than $1.5 * \pi$?" She smiled, continuing, "I'll bet that a program designed to take advantage of that knowledge, which lets the computer skip all that computing, might just plot a little faster."

The elegance of a one-liner was lost on her. Okay, here are two FOR...NEXT loops to replace the testing: one to $1.5 * \pi$, the other from there to $2.5 * \pi$. YANG/YIN-4 was the result, and ran in 20 seconds. "Great!" The problem had seen its solution now.

My beloved's only comment was, "I like circles because they're so symmetrical. Put a mirror down the center line of half a circle, and it will appear whole — the reflection just like the image."

Sometimes I don't understand her at all — mirror images now! Then, sometimes I do follow her intuitive jumps. In this case, it was apparent that half my figure could be plotted as a mirror image, if one used the $X = 140$ line for mirror placement. When T is between .5 and $1.5 * \pi$, $\cos(T)$ is a negative value which makes $140 + X$ less than the baseline $X = 140$. When T is greater than $1.5 * \pi$, the values derived are similar, but positive. Subtracting a negative is the equivalent of adding a positive...hmmm! Let's try it by employing one FOR...NEXT loop to $1.5 * \pi$, one H PLOT using $140 + X$, and one H PLOT using $140 - X$.

YANG/YIN-5 is the end result, a one-line program which runs in about 13 seconds, and is close to minimum for the particular solution to this plotting problem. With 33 operations on 41 called variables, and 100 iterations, a mini/max balance has been achieved so that a further sectioning of the circle increases the variables and operations to the point where speed is actually lost. Fewer iterations via larger STEPping-say, a halving by using STEP

$2 * \pi / 100$ -produces a rather unesthetically pleasing figure with a lot of lines showing in the light, or "yang" side.

Care must be taken in entering YANG/YIN-5, for it really is too long for a one-line program. All spaces must be eliminated, and the PRINT statement tokenized with the ? substitute. If an error is made, corrections can only be made with a line editor, or by retyping the program in its entirety. The following is a brief explanation of the various parts of the program:

```
10 HOME : HGR : HCOLOR = 7
sets up HIRES graphics in white
:K = 2: A = 60: E = 1.2 sets
variables serving as constants
:Z = 140: L = 80 sets baselines
:M = 110: N = 50 sets plotting
centers
:P = 3.1416: B = 60 sets pi,
initializes variable B for 1st plot Y, C
is 0 automatically
:FOR T = P/K TO K * P sets loop
for 100 iterations
STEP K * P / 200 STEPping every 1.8
degrees
:X = A * E * COS ( T ) X axis
modifying variable
:Y = A * SIN ( T ) Y axis modifying
variable
:H PLOT Z + X, L + Y left side
yang start plot
TO Z + X / K, M + Y / K left side
yang finish plot
:H PLOT Z - X / K, N + Y / K
right side yang start plot
TO Z - X / K, N - Y / K right side
yang finish plot
:H PLOT Z - C, L + B right side
perimeter old point
TO Z - X, L + Y right side perimeter
new point
:C = X: B = Y remember the last
plot point
:NEXT : V TAB 22 end of loop, set
line to print on
:PRINT TAB (17) "YANG/YIN"
prints program name
```

All these parts can be set up in their own lines and modified by experimenters eager to beat the clock. I suspect that a really significant bettering of the execution time will require a different algorithm for deriving the points to be plotted.

My wife has begun to wonder what language the Apple II really "likes." I wrote a little program for her as a punishment for being so hard to please (see ETI-YO in Table 1), but only in fun.

"Isn't that last word misspelled, Dear?"...

Table 1
TIMER SUBROUTINE

```

1 GOSUB 1000:T1 = T
2 REM ###>> INSERT THE PROGRAM
3 REM ###>> TO BE TIMED BY THIS
4 REM ###>> TIMING SUBROUTINE
5 REM ###>> ANYWHERE, FROM HERE
995 REM ###>> TO HERE!
997 GOSUB 1000:T2 = T
998 T = T2 - T1: PRINT "TIME = ";
T
999 END
1000 PRINT CHR$(4);"IN04": INPUT
A$: PRINT CHR$(4);"IN00":T
= VAL ( MID$( A$,13,6) ) +
VAL ( MID$( A$,10,2) ) * 60:
RETURN

```

YANG/YIN-1

```

10 HOME : HGR : HCOLOR= 7
14 R = 60:C = 80
16 FOR T = .5 * 3.1416 TO 2.5 *
3.1416 STEP 2 * 3.1416 / 200

18 X = 140 + 1.2 * R * COS (T)
20 Y = C + R * SIN (T)
22 IF T = .5 * 3.1416 THEN HPLLOT
X,Y
24 IF TC THEN HPLLOT X,Y:TC = 0
26 IF T < > 0 THEN HPLLOT TO X
,Y
28 IF C = 110 AND T > 1.5 * 3.14
16 THEN C = 50:TC = 1
30 NEXT
32 IF R = 60 THEN R = 30:C = 110
: GOTO 16
34 VTAB 22: PRINT TAB( 16)"YANG
/YIN"

```

YANG/YIN-2

```

10 HGR : HCOLOR= 7: FOR T = .5 *
3.1416 TO 2.5 * 3.1416 STEP
2 * 3.1416 / 200
20 X = 1.2 * 60 * COS (T):Y = 60
* SIN (T)
30 IF T < 1.5 * 3.1416 THEN HPLLOT
140 + X,80 + Y TO 140 + X /
2,110 + Y / 2
32 IF T > 1.5 * 3.1416 THEN HPLLOT
140 + X / 2,50 + Y / 2 TO 14
0 + X / 2,50 - Y / 2
34 IF T = .5 * 3.1416 THEN X2 =
X:Y2 = Y
36 HPLLOT 140 + X2,80 + Y2 TO 140
+ X,80 + Y
38 X2 = X:Y2 = Y
40 NEXT

```

YANG/YIN-3

```

10 P = 3.1416:Y2 = 60:Z = 140: HGR
: HCOLOR= 7:V = 1.5 * P: FOR
T = .5 * P TO 2.5 * P STEP 2
* P / 200:X = 1.2 * 60 * COS
(T):Y = 60 * SIN (T): HPLLOT
Z + X / (1 + 1 * (T > V)),50
+ 30 * (T < V) + Y / (1 + 1
* (T > V)) TO Z + X / 2,50 +
60 * (T < V) + Y / 2 - Y * (
T > V): HPLLOT Z + X2,80 + Y2
TO Z + X,80 + Y:X2 = X:Y2 =
Y: NEXT : VTAB 22: PRINT "YA
NG/YIN"

```

YANG/YIN-4

```

10 HOME : HGR : HCOLOR= 7:P = 3.
1416:Y2 = 60:Z = 140:V = 1.5
* P: FOR T = .5 * P TO V STEP
2 * P / 200:X = 1.2 * Y2 * COS
(T):Y = Y2 * SIN (T): HPLLOT
Z + X,80 + Y TO Z + X / 2,11
0 + Y / 2: NEXT :Y2 = Y:X2 =
X
20 FOR T = V TO 2.5 * P STEP 2 *
P / 200:X = 1.2 * 60 * COS
(T):Y = 60 * SIN (T): HPLLOT
Z + X / 2,50 + Y / 2 TO Z +
X / 2,50 - Y / 2: HPLLOT Z +
X2,80 + Y2 TO Z + X,80 + Y:
X2 = X:Y2 = Y: NEXT : VTAB 22
: PRINT TAB( 16)"YANG/YIN"

```

YANG/YIN-5

```

10 HOME : HGR : HCOLOR= 7:K = 2:
A = 60:E = 1.2:Z = 140:L = 8
0:M = 110:N = 50:P = 3.1416:
B = 60: FOR T = P / K TO 1.5
* P STEP K * P / 200:X = A *
E * COS (T):Y = A * SIN (T
): HPLLOT Z + X,L + Y TO Z +
X / K,M + Y / K: HPLLOT Z - X
/ K,N + Y / K TO Z - X / K,
N - Y / K: HPLLOT Z - C,L + B
TO Z - X,L + Y:C = X:B = Y:
NEXT : VTAB 22: PRINT TAB(
17)"YANG/YIN"

```

ETI-YO

```

10 HOME : HGR : HCOLOR= 7:A = 60
:E = 1.2:L = 140:M = 80:B =
1.9:P = 3.1416:K = 2: FOR T =
0 TO P STEP K * P / 200:X =
A * E * COS (T):Y = A * SIN
(T): HPLLOT L + X,M + Y TO L +
B * X,M + Y: HPLLOT L + X,M -
Y TO L + B * X,M - Y: NEXT :
VTAB 22: PRINT TAB( 5)"YOU
ARE THE APPLE OF MY EYE!!!"

```

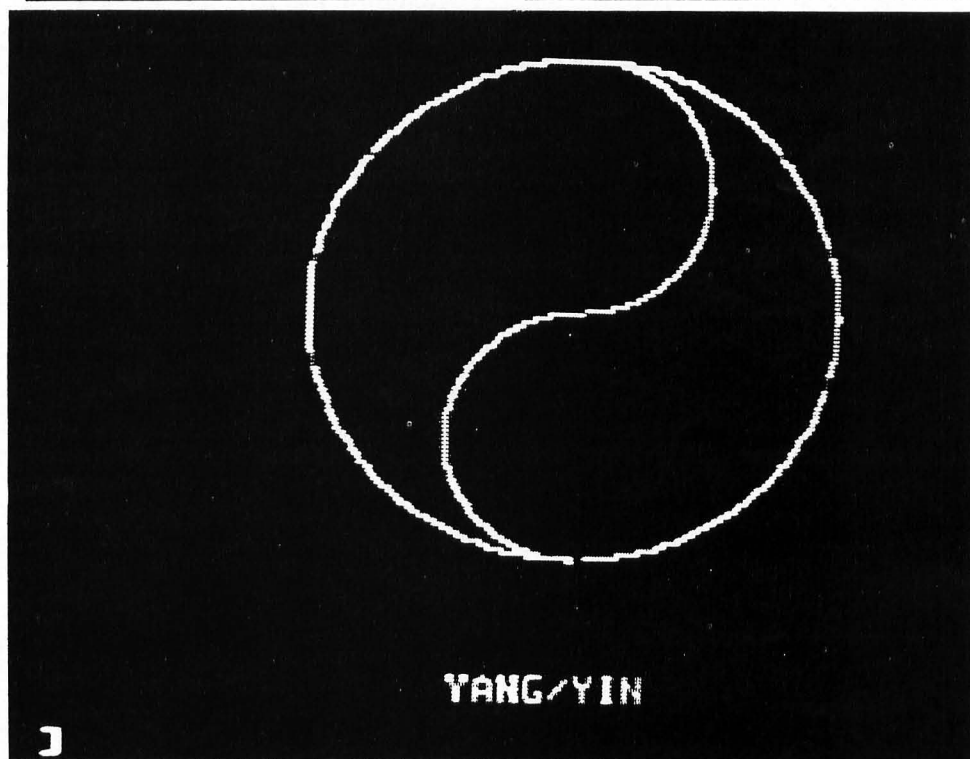


Figure 1



Take Apart: Atari Quest

by Alan J. Zett

If you didn't see the program "Quest 1" in the August 1981 issue, you are missing out on one of the best programs **SoftSide** has ever published. But, if you did see "Quest 1," and you own an Atari, you've probably been asking yourself a lot of questions while hunting through the program code or even muttering under your breath because of the special "tricks" it uses that you might not fully understand.

When I wrote the Atari translation, I felt that this program was good enough to warrant some special attention to hidden Atari features. I decided to exclude sound because it would detract from the game. We've all heard of the untold mysteries of the Atari 800, but few have discovered what they really are.

I first redefined the player's game character (usually an "@" symbol) to look like a man with a raised sword. I was pretty much satisfied with this until I recalled reading an article in the October 1980 issue of **SoftSide** on how the Atari, while in the graphics mode, will plot a yellow dot next to a blue dot to get a white dot. I decided to try this on the character set and it worked.

I reasoned that it should be possible to redefine a character to be an inverse space (a white block) in a color of my choosing. It also turns out that when this character is inversed, it will become the complementary color of itself. Using this idea, I constructed the dungeon display with what appeared to be color graphics surrounded by text, when in actuality the display was completely made up of multi-colored text characters.

Let me explain in more detail how this display was done. First, there are some important memory locations for you to learn listed in Figure 1. They are integral to the understanding and operation of my concept, so study them carefully.

Because the character set is in ROM, we must move it to RAM in order to modify it. (ROM stands for Read Only Memory; we can never write to it, it is permanent. RAM means Random Access Memory; we can do whatever we like with it.) Fortunately, this move isn't very difficult. We must PEEK at the characters in the ROM and then POKE them into our new location in RAM, being sure to clear the area first

and pointing BASIC to the new set after they've all been moved. This is accomplished by the following lines:

```
5 REM CHARACTER SET IS 1K
BYTES LONG SO BUMP THE END
OF MEMORY BACK 1K (4 PAGES)
+ 1 PAGE FOR GOOD MEASURE.
CLEAR THAT AREA WITH A
GRAPHICS 0 COMMAND TO
READJUST THE BASIC
POINTERS TO THE NEW RAM
TOP.
10 POKE 106, PEEK (106)-5:
GRAPHICS 0
15 REM GET THE POSITION
WHERE THE NEW CHARACTER
SET WILL BE LOCATED INTO
THE VARIABLE START AND
MOVE THE OLD CHARACTER
SET THERE. THIS WILL TAKE
ABOUT 22 SECONDS.
20 START=(PEEK(106)+1)*256:
FOR X=0 TO 1023: POKE
START+X, PEEK(57344+X):
NEXT X
```

There! The character set is now in RAM and we are almost ready to begin modifying it. But before we can do that, you'll have to find out more about the way ROM stores a character.

Refer to Figure 2 to see how the ATASCII codes compare to their position in RAM. As you can see, codes

normally occupying the values between 0 and 31 are graphics characters, but in the ROM, they are located between 64 and 95. Codes between 32 and 95 are located from 0 to 63 and those from 96 to 127 are the same. Codes above 127 are inverse codes of the first 128 characters. For example: a normal uppercase "A" is character number 65. An INVERSE uppercase "A" is 65 + 128 or 193.

Each character consists of 8 "bit-mapped" bytes. Bit-mapped simply means that each of the 8 bits in the byte stand for something different. In this case, 8 bytes can form an 8 by 8 array of bits. Since a bit can only be a "0" or a "1," a "0" indicates a dot in the character that will not be lit and a "1" indicates a dot that will. When drawn together, these dots give us an 8 by 8 dot matrix character on the screen. If we want to change a character, all we need to do is change the bits that we want on to "1's" and those we want off to "0's."

Each bit is assigned a value so that when all 8 bits are "on" or "1," the total will be 255. We don't have to get too involved in binary numbers; take my word for it when I tell you that Figure 3 shows the value assigned to

Figure 1:

Table of important memory locations

Decimal	Hexidecimal	Purpose
00106	006AH	Holds the value of the last free page (256 bytes) of memory in RAM.
00756	02F4H	Holds the value of the starting page in memory where the character set resides.
57344	E000H	Location of the original character set in the ROM Operating System.

Figure 2:

The position in memory of a character based on its ATASCII code.

ATASCII Value (X) Memory Location in RAM

000 to 031 (X + 64) * 8 + START
032 to 095 (X - 32) * 8 + START
096 to 127 X * 8 + START

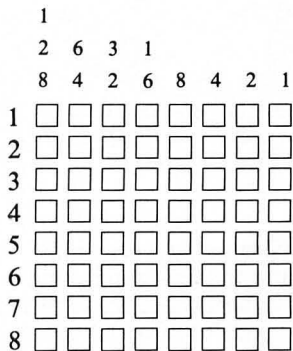
Figure 3:

Values of bits depending on their positions.

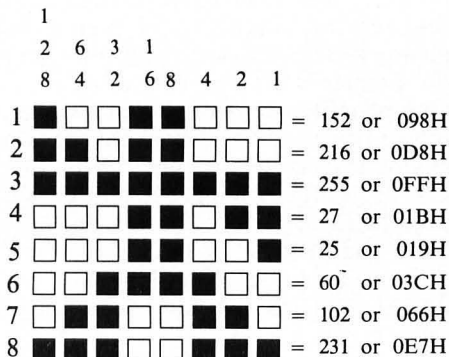
Relative Bit Position	Multiply Bit Value by:
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

each of the 8 bits in our byte.

If we take a little graph paper and mark off an 8 by 8 grid, we can now draw the shape we wish to use. The graph should look something like this:



The numbers on the left side indicate the number of the byte. The boxes indicate the row of 8 bits for that byte (all set to "0's") and the numbers above them indicate the value of the bit at that position. If I want to draw a man character, the grid would look like this (filled boxes indicate on bits and empty ones indicate off bits):



Now that we have the data we need, all we have to do is look up the character we want to change (in this case, the "@" symbol) in Figure 2 based on its ATASCII value and we come up with (64 - 32) * 8 + START or 256 + START. This is the location of the 8 bytes that make up the "@" symbol. Now all we have to do is a short 2 line routine to change it as follows:

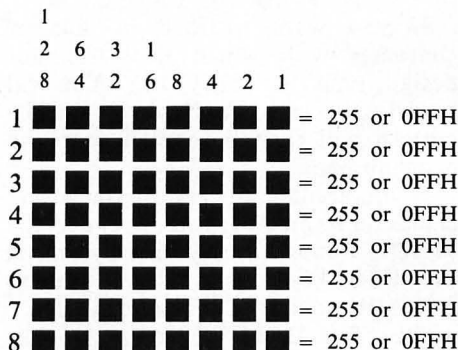
```
30 RESTORE: FOR X=0 TO 7:
READ Y: POKE X + 256 + START, Y:
NEXT X
40 DATA 152, 216, 255, 27, 25, 60,
102, 231
```

We now have an "@" that looks like a man. But, you say, he's WHITE. HOW DID YOU GET THE COLORED WALLS IN "QUEST"?

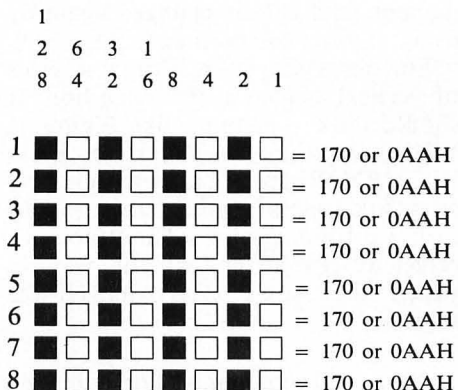
Unfortunately, the answer to that requires a bit more thought. I

mentioned earlier that BASIC will plot a yellow dot next to a blue dot to get a white dot. If you take this idea a step further, you can begin to see where I'm heading.

If we were to draw a single dot in a character without another dot right next to it, we will see a non-white single color. This color is mainly dependent on the BACKGROUND color used, not the text color as you would suspect. For instance, take the following character:

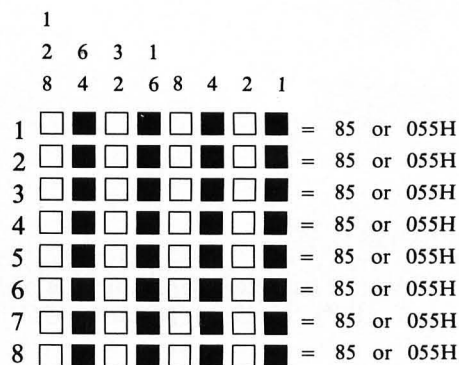


This is the "bit-map" of the inverse space. Note how every bit is used and that every bit has another bit next to it. If we modify it like this:



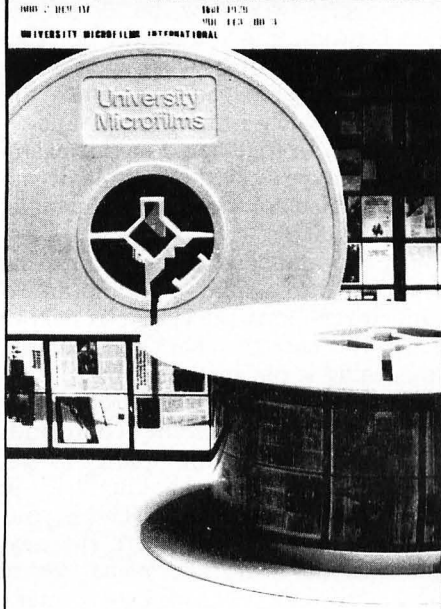
It will still look like a full block (because a standard TV picture tube can't separate lines that close without smearing them together), but this block is now colored, as opposed to white. Furthermore, by using the inverse key, we can have another color (the first color's natural inverse: e.g., red is the inverse of green) giving us a total of 4 colors in the text mode: white, the background color, and 2 other colors dependent on the background color. Or you could look at it like this: white text, and 4-color block graphics.

All the inverse key does to the block is reverse all the on bits in a character to off bits and vice versa. The character used in our example, when inverted, will look like this:



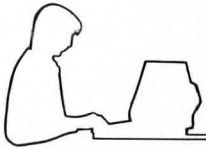
If we add this line:
50 FOR X=0 TO 7: POKE
X+48+START, 170: NEXT X
this will give us the colored block pattern (and thus our other 2 colors) redefined for the "&" character. ☺

This publication is available in microform.



**University Microfilms
International**

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



Character Generator

by Alan J. Zett

“Atari Character Generator” is an Atari utility program requiring 16K RAM.

The following are instructions and documentation for the Character Generator. For a more in-depth explanation, see “TAKE APART — Atari Quest” elsewhere in this issue.

First, press SYSTEM RESET, load the program, and type RUN. It is absolutely necessary to press SYSTEM RESET prior to RUNNING any program using a modified character set. Forgetting to do so can cause the computer to lock-up and lose the program currently in memory.

After about twenty seconds you should see all the characters (except inverse) displayed on the left side of the screen, a large box displayed on the right side, and at the bottom the message “Edit character or Save file?”. Let’s assume for now that you type “E.”

The message “X,Y coordinate of the character?” will now be displayed. This is prompting you for the horizontal (X) position of the character, which is between 0 and F, and the vertical (Y) position, which is between 0 and 7. Type them in together. For this example we’ll type 0,2 (the comma is optional) which corresponds to the blank space character.

The computer will now put that character into a small box, and an expanded version in the large box next to it. (In this case the character is blank, so nothing is put in either box.)

The message at the bottom shows you the nine different commands available to you in the modify mode. You should also see a blinking “+” in the large box on the right. This is your cursor. It shows you where the next modification will take place. Note: Only DELETE and RETURN make actual changes to the character set as stored in memory; the other commands affect only the display.

To plot a block in the character, use the SET command. Use the arrow keys to position the cursor where you want to make a change, and press “S” to turn the block on. To do the opposite, type “R” to RESET the block, positioning the cursor in the same way as for the SET command. If you want to

erase the entire drawing pad, type “E” for ERASE. Or, to start with all blocks on, type “W” to WHITE out all blocks. Reversing the drawing is done by typing “I” for INVERSE.

If you want to load in another character with which to start your design, type “C” for COPY. You will be asked to select a character, and its pattern will then be copied onto the drawing pad.

If you accidentally choose the wrong character to modify, you can cancel the MODIFY mode at any time by hitting the ESC key. This allows you to ESCAPE to the “X,Y of character” mode. The DELETE command lets you change your mind about a modification by restoring the current character to its unmodified form.

Finally, when everything is correct, hit RETURN to make the change both in memory and on the screen. Remember that this change is not permanent, and can be changed again by using any of the commands.

For our example, let’s draw a series of vertical stripes in the large box. It should look something like Figure 1. When everything is just right, press RETURN. Right in front of your eyes, all blank spaces on the screen should turn a different color. (The “fade-in” effect is reminiscent of the way in which the Apple clears its Hi-Res screen.)

You can keep on modifying all night if you want, but for now, we’ll call it a day. When you see the message “X,Y of ...,” press RETURN again. This will take you back to the original menu. From here, select the “S” option to save the file. The computer will ask, “Want to SAVE (Y/N) ?” In case you hit “S” by accident, type N. Otherwise, type Y and hit RETURN. The prompt, “To Cassette or Disk?” will be displayed; answer with a “C” or “D”.

If you said “D”, then answer the next question with the 8-letter filename plus a 3-letter extension, all preceded by a “D:”. Answer the next question “Is this right?” with a “Y” or “N” and hit RETURN.

If you said “C”, for a cassette save, then you’ll hear two beeps. Prepare the cassette and press RETURN. After the

save, type “Y” to continue or “N” to stop.

Now that you’ve saved it, what can you do? This is the best part! Write your program between lines 2 and 31999 using the characters that will be modified where necessary. When it’s all done and totally debugged, type either ENTER “C” for cassette or ENTER “D:filename.ext” for disk. This will merge the custom character set into your program, and it should run perfectly the first time.

Two final notes: First, the new program will take an extra 1.25K of memory and will take longer to initialize. Second, after every GRAPHICS command you must insert a POKE 756, PEEK (106)+1. This is because the GRAPHICS command restores the character set pointer in memory. The new set is still there, but the pointer must be reset to access it. One benefit from this is that you can toggle back and forth between the two character sets by typing POKE 756, 224 to get the original set and POKE 756, PEEK (106)+1 for the new set. Note that PRINT CHR\$(125) clears the screen as does GRAPHICS 0, but does not affect the current character set.

Well, that’s it. I hope you enjoy yourself! We’ll be looking forward to new Atari programs using custom character sets in future issues of **SoftSide**.

Variables

A\$: Misc. string input.
B: Binary bit value counter.
B(*): Modified character pointer array (-1 = not modified).
C: Character byte value.
CC: Current character at cursor position.
CH: Command character.
CU: Cursor character.
CX: Cursor X position.
CY: Cursor Y position.
FL: Modify flag.
HEX\$: Used in converting hex input to decimal.
KP: Key pressed? (No = 255.)
START: Starting location in memory of new character set.
W, X, Y, Z: Misc.

Initialization.

```
10 OPEN #1,4,0,"K":OPEN #2,12,0,"S"
12 CLR :DIM B(127),MEX$(16)
14 DIM A$(14):GOSUB 10000
20 MEX$="0123456789ABCDEF"
22 SETCOLOR 2,9,2:SETCOLOR 4,4,4
24 POKE 82,21
```

Redraw entire display.

```
30 ? CHR$(125):POKE 752,1
32 POSITION 19,0
33 ? " CHARACTER GENERATOR "
34 POSITION 19,1
35 ? " BY ALAN J. ZETT "
36 FOR X=7 TO 16 STEP 9:POSITION 2,X
38 ? " 0123456789ABCDEF ";NEXT X
40 FOR X=8 TO 15:POSITION 2,X
42 ? CHR$(X+168):POSITION 19,X
44 ? CHR$(X+168):NEXT X
50 POSITION 4,5: ? "Character set:";
60 POKE 766,1:FOR X=0 TO 7
62 FOR Y=0 TO 15:POSITION Y+3,X+8
64 ? CHR$(X+16+Y):NEXT Y:NEXT X
66 POKE 766,0:POKE 752,1
70 POSITION 26,7: ? " ";
71 REN <CTRL> 'RRRRRRRRR'
72 POSITION 26,16: ? " ";
73 REN <CTRL> 'XXXXXXXXXX'
80 FOR X=8 TO 15:POSITION 26,X
82 ? " ";NEXT X
83 REN <SHIFT> ' = '
90 POSITION 24,14: ? " ";
91 REN <CTRL> 'GRD'
92 POSITION 24,15: ? " ";
93 REN <SHIFT> ' = '
94 POSITION 24,16: ? " ";
95 REN <CTRL> 'ZRX'
```

Select main program options.

```
100 GOSUB 9000:POSITION 2,20
102 POKE 752,0
104 ? "Edit character or Save file";
106 INPUT A$
108 IF A$(1,1)="S" THEN 570
110 IF A$(1,1)<>"E" THEN 100
```

Input and adjust value of character.

```
120 GOSUB 9000:POSITION 2,20:TRAP 66
122 ? "X,Y coordinate of the character
";:INPUT A$
130 IF A$(LEN(A$))<"0" OR A$(LEN(A$))>
"7" THEN 120
140 TRAP 3333:Y=VAL(A$(LEN(A$))):X=1
150 IF A$(1,1)<>MEX$(X,X) THEN X=X+1:IF
X<17 THEN 150
160 X=X-1:IF X>15 THEN 120
170 POKE 752,1:Z=X+Y*816
172 POSITION 25,15:POKE 766,1
```

```
174 ? CHR$(Z):POKE 766,0
180 IF Z<32 THEN Z=Z+64:GOTO 200
190 IF Z<96 THEN Z=Z-32
```

Draw character onto pad.

```
200 POKE 82,27
210 FOR Y=0 TO 7:X=0:C=PEEK(START+Y+Z*
8):B=256
220 B=B/2:IF B<1 THEN 270
230 POSITION X+27,Y+8:X=X+1
240 IF (C-B)<0 THEN ? " ";:GOTO 260
250 IF (C-B)>=0 THEN ? " ";:C=C-B
260 GOTO 220
270 NEXT Y
```

Obtain modify command and branch.

```
280 GOSUB 9000:POSITION 2,20:POKE 82,2
290 POKE 752,1: ? "Copy, Delete, Erase,
Inverse, Reset,"; ? " ? "Set, White, ES
Cape, RETURN to store.";
300 POKE 752,1:POKE 764,255:CX=27:CY=8
310 POSITION CX,CY:GET #2,CC
320 POSITION CX,CY:PUT #2,CC:KP=PEEK(7
64):CU=43:IF CC<>32 THEN CU=171
330 IF KP=255 THEN POSITION CX,CY:PUT
#2,CU:GOTO 320
340 GET #1,CH:IF CH=155 THEN 470
350 IF CH=27 THEN 120
352 IF CH=ASC("E") THEN 790
354 IF CH=ASC("C") THEN 820
356 IF CH=ASC("D") THEN 810
357 IF CH=ASC("W") THEN 1000
358 IF CH=ASC("I") THEN 1020
360 IF CH=ASC("-") THEN CY=CY-1
370 IF CH=ASC("=") THEN CY=CY+1
380 IF CH=ASC("+") THEN CX=CX-1
390 IF CH=ASC("#") THEN CX=CX+1
400 IF CH=ASC("S") THEN CC=160:GOTO 32
0
410 IF CH=ASC("R") THEN CC=32:GOTO 320
```

Correct cursor position in pad.

```
420 IF CX<27 THEN CX=34
430 IF CX>34 THEN CX=27
440 IF CY<8 THEN CY=15
450 IF CY>15 THEN CY=8
460 GOTO 310
470 POSITION CX,CY:PUT #2,CC
```

Make character modifications.

```
480 FOR Y=0 TO 7:X=0:B=256:CC=X:C=X
490 B=B/2:IF B<1 THEN 540
500 POSITION X+27,Y+8:X=X+1:GET #2,CC
510 IF CC=32 THEN 530
520 IF CC=160 THEN C=C+B
530 GOTO 490
540 POKE START+Y+Z*8,C
```

```
550 NEXT Y:POSITION 34,15
560 PUT #2,CC:B(Z)=Z*8
562 FL=0:FOR Y=0 TO 7
564 IF PEEK(START+Y+Z*8)<>PEEK(57344+Y
+Z*8) THEN FL=1
566 NEXT Y:IF FL=0 THEN B(Z)=-1
568 GOTO 120
```

Save file routine. Note: Be especially careful typing these lines. Correct spacing is very important.

```
570 PRINT CHR$(125):POKE 82,2
580 TRAP 990:POSITION 2,5
582 ? "You want to SAVE (Y/N)";
584 INPUT A$:IF A$(1,1)<>"Y" THEN 30
590 ? "To Cassette or Disk";:INPUT A$
600 IF A$<>"D" THEN 620
610 ? "Enter file 'D:FILENAME.EXT'"
612 INPUT A$: ? "Is '";A$;"' correct ? "
;
614 GET #1,CH:IF CH=78 THEN 610
620 OPEN #3,8,0,A$
630 PRINT #3;"1 GOSUB 32000:CLR"
640 PRINT #3;"32000 POKE 106,PEEK(106)
-5:GRAPHICS 0:START=(PEEK(106)+1)*256:
POKE 756,START/256:POKE 752,1"
650 PRINT #3;"32010 ? ";CHR$(34):"INIT
IALIZING . . .";CHR$(34)
660 PRINT #3;"32020 FOR Z=0 TO 1023:PO
KE START+Z,PEEK(57344+Z):NEXT Z:RESTOR
E 32100"
670 PRINT #3;"32030 READ X:IF X=-1 THE
N RESTORE :RETURN"
680 PRINT #3;"32040 FOR Y=0 TO 7:READ
Z:POKE X+Y+START,Z:NEXT Y:GOTO 32030"
690 LN=32100:LI=1:FOR Z=0 TO 127
700 IF B(Z)=-1 THEN 750
710 PRINT #3;STR$(LN):" DATA ";
720 PRINT #3;STR$(B(Z));
730 FOR Y=0 TO 7:PRINT #3:",";STR$(PEE
K(START+Y+Z*8)):NEXT Y
740 PRINT #3:",";LN=LN+LI
750 NEXT Z:PRINT #3;STR$(LN):" DATA ";
STR$(-1):CLOSE #3:TRAP 33333
760 POKE 764,255: ? "WANT TO CONTINUE";
:INPUT A$
770 IF A$(1,1)="Y" THEN 30
780 ? CHR$(125):END
```

Erase command.

```
790 FOR Y=8 TO 15:POSITION 27,Y
800 ? " ";:NEXT Y:GOTO 300
```

Delete command.

```
810 B(Z)=-1:FOR Y=0 TO 7:POKE START+Y+
Z*8,PEEK(57344+Y+Z*8):NEXT Y:GOTO 120
```

continued on page 83



OCTOBER ADVENTURE OF THE MONTH CRIME ADVENTURE

Test your skills as a detective, sifting through hundreds of clues. You may have to become the new Sherlock Holmes to solve this one!

A new obsession is sweeping the country as members of **SoftSide's Adventure of the Month Club** rush to their mailboxes each month. Where will they be taken this month — Off into space? Back in time? To the bottom of the sea?

You too can look forward to an exciting new spree, full of secrets, every month. Your program budget won't be destroyed, either. Membership in **SoftSide's Adventure of the Month Club** costs only \$27 for six months on tape (\$4.50 per adventure) or \$45 for six months on disk (\$7.50 per adventure).

You can sample these delights with one month's edition. The tape is only \$6 and the disk \$9.

To join this unique crew, use the order card bound into this issue. Fill it out and send with payment to:

Adventure of the Month Club
Department 681
6 South Street
Milford, NH 03055

Sit back and wait for your first program to arrive. **SoftSide** will select a high quality, original Adventure in BASIC. Watch for the announcement of each month's Adventure here in the pages of **SoftSide Magazine**.

SYSTEM REQUIREMENTS:

Apple 24K Tape 32K Disk
Atari 32K Tape 40K Disk
S-80 16K Tape 32K Disk

THIS MONTH'S SPECIAL:

SmartWare (LISP) for the Apple II from
DataSoft T.M.

The facts: Relational database capabilities. User & program definability. Pattern-directed invocation language. User syntax & data structures upon which esoteric applications may be implemented. Remembers data along with "relationships affecting it." At MIT, they call LISP the language of the future. (See the SoftSide review - Aug. 81 issue, pg. 92).

48K DISK, APPLE II, REG. \$125.00

SPECIAL
\$112.50

offer ends Nov. 15th!

SoftSide
Selections
6 South Street Milford NH 03055

SoftSide Selections Order Form

S-80 PROGRAMS

- Alien Invasion,
16K Tape, Machine Language \$9.95
- Alien Invasion,
32K Disk, Machine Language \$14.95
- Column Calculator 4.1,
32K Disk (Mod I only) \$39.95
- End Zone II, 16K Tape \$14.95
- End Zone II, 32K Tape \$15.95
- Floppy Disk Diagnostic,
16-48K Disk (Mod I only) ~~\$24.95~~ \$22.50*
- Nine Games for Preschool Children,
16K Tape \$9.95
- Orchestra 80,
16K Software/Hardware combo ~~\$80.00~~ \$69.95*
- Star Trek III.5, 16K Tape \$14.95
- Star Trek III.5, 32K Disk \$15.95
- Typing Tutor, 16K Tape \$19.95
- Typing Tutor, 32K Disk \$20.95
- World Series, 16K Tape \$9.95

APPLE PROGRAMS

- Pool 1.5, 48K Disk ~~\$34.95~~ \$29.95*
- SmartWare (LISP) ~~\$125.00~~ \$112.50*
- Time Lord, 48K Disk, Integer \$29.95
- World Series, 16K Tape \$9.95
- World Series, 32K Disk \$14.95

ATARI PROGRAMS

- Masters' Golf, 8K Tape \$9.95
- Star Trek III.5, 32K Tape \$19.95
- World Series, 16K Tape \$9.95

OTHER

- The Apple II User's Guide \$15.00
- The Basic Handbook ~~\$19.95~~ \$17.95*
- Vinyl Binder (small) \$3.95
- Vinyl Binder (large) \$7.95

*Offer Expires November 15.

ADVENTURE OF THE MONTH

- Crime Adventure Tape \$6.00
for the _____ computer
- Crime Adventure Disk \$9.00
for the _____ computer

ADVENTURE OF THE MONTH CLUB

- 6 month Tape subscription \$27.00
for the _____ computer
- 6 month Disk subscription \$45.00
for the _____ computer

SOFTSIDE SUBSCRIPTIONS

- Magazine only (12 issues) \$24.00

With the **SoftSide** cassette subscription you receive not only the magazine, but all of the programs in it on tape. The Enhanced Disk Version includes the magazine, its programs on disk, AND additional programs only documented in the magazine.

- Magazine and cassette (12 issues) \$75.00
- Magazine and enhanced disk (12 issues) \$125.00
- Sample Magazine and Enhanced Disk
(October) \$19.95

Enter my subscription for the Apple Atari S-80 computer.

Please use facing bind-in card to order. If card is missing, be sure to add \$1.50 handling charge to your order total.

Canada/Mexico Orders

No C.O.D. to Canada or Mexico. The preferred method of payment is by Master Card 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 Master Card. All shipping and duty charges are the customer's responsibility. All overseas orders are subject to a \$10.00 handling charge PLUS actual shipping 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) 673-0585 or 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 of operation of such software.

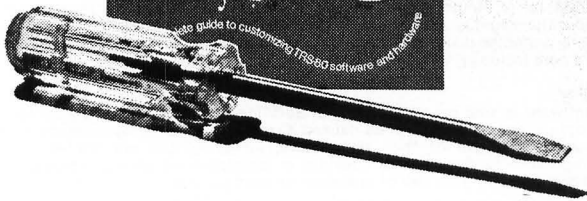
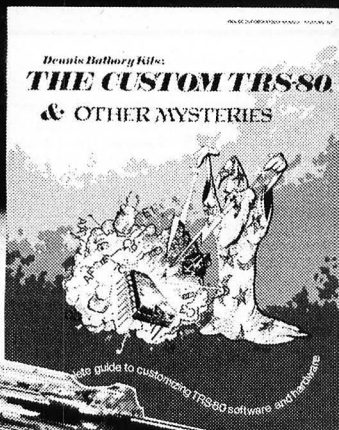
Prices:

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



New Computer Book

TUNE-UP YOUR TRS-80®



Ever wanted to do things to your TRS-80 that Radio Shack said couldn't be done? How about reverse video, high resolution graphics, a high-speed clock, and audible keystrokes?

Not enough? How about turning an 8-track into a mass storage device, making music, controlling a synthesizer, individual reverse characters, and a real-time clock?

If the thought of using a screwdriver gives you the shivers then you can turn to the software section. Learn how to make BASIC programs auto-execute, reset memory size, pack program lines with machine code, and generate sound effects.

The **Custom TRS-80 and Other Mysteries** is packed with page after page of practical information, and tested software. Get a copy and turn your TRS-80 into a supercomputer.

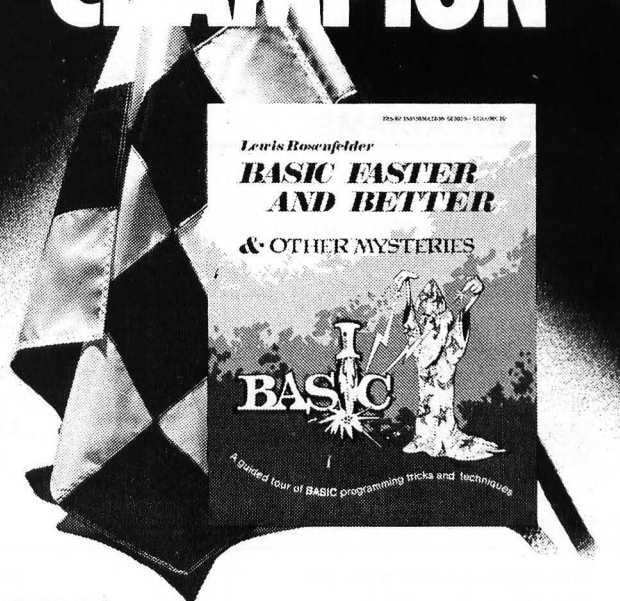
#65-238001B \$29.95

MasterCard VISA

TSE-HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

New Computer Book

MAKE BASIC PERFORM LIKE A CHAMPION



BASIC is not nearly as slow as most programmers think. **BASIC Faster and Better** shows you how to supercharge your BASIC with 300 pages of fast functions and subroutines.

You won't find any trivial, poorly designed "check-book balancing" programs in this book — it's packed with useful programs.

Tutorial for the beginner, instructive for the advanced, and invaluable for the professional, this book doesn't just talk... it shows how! All routines are also available on disk, so that you can save hours of keyboarding and debugging.

The #1 disk, BFBDEM, contains all the demonstration programs, and #2 BFBLIB, has all the library functions.

#65-238004B \$29.95

#1 Disk (BFBDEM)

#25-238005T \$19.95

#2 Disk (BFBLIB)

#25-238006T \$19.95

MasterCard VISA

TSE-HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

continued from page 79

Copy command.

```

820 GOSUB 9000:POSITION 2,20
822 ? "X,Y of character to copy";
824 INPUT A$
830 IF A$(LEN(A$))<"0" OR A$(LEN(A$))>
"7" THEN 820
840 POKE 752,1:Y=VAL(A$(LEN(A$))):X=1
850 IF A$(1,1)<>HEX$(X,X) THEN X=X+1:1
F X<17 THEN 850
860 X=X-1:IF X>15 THEN 820
870 W=X+Y*16
880 IF W<32 THEN W=W+64:GOTO 900
890 IF W<96 THEN W=W-32
900 FOR Y=0 TO 7
910 X=0:C=PEEK(START+Y+W*8):B=256
920 B=B/2:IF B<1 THEN 970
930 POSITION X+27,Y+B:X=X+1
940 IF (C-B)<0 THEN ? " ";GOTO 960
950 IF (C-B)>=0 THEN ? " _";C=C-B
960 GOTO 920
970 NEXT Y
980 GOTO 280

```

Error trap.

```

990 ? " INPUT ERROR, RE-ENTER. ";GOTO
582

```

White command.

```

1000 FOR Y=8 TO 15:POSITION 27,Y
1010 ? " _____";NEXT Y:GOTO 300

```

Inverse command.

```

1020 FOR Y=8 TO 15:FOR X=27 TO 34
1030 POSITION X,Y:GET #2,CH:POSITION X
,Y
1040 IF CH=32 THEN PUT #2,160
1050 IF CH=160 THEN PUT #2,32
1060 NEXT X:NEXT Y:GOTO 300

```

Erase old messages.

```

9000 POKE 752,1:POSITION 0,20
9010 ? " ";POKE 752,0:RETURN
9020 REM 5 (ESC)<SHIFT><DELETE>'8

```

Download new character set and reset modify pointers to -1.

```

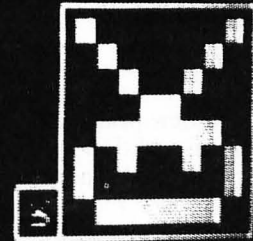
10000 POKE 106,PEEK(106)-5:GRAPHICS 0
10010 START=(PEEK(106)+1)*256
10020 POKE 756,START/256:POKE 752,1
10030 ? "INITIALIZING . . .":? " (TA
KES ABOUT 20 SECONDS)"
10040 FOR Z=0 TO 1023
10050 POKE Z+START,PEEK(Z+57344)
10060 NEXT Z:FOR Z=0 TO 127
10070 B(Z)=-1:NEXT Z:RETURN

```



CHARACTER GENERATOR
BY ALAN J. ZETT

character set:

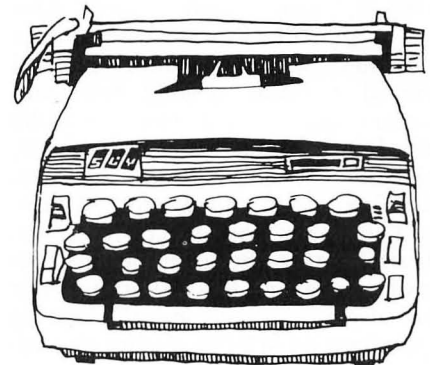


Y coordinate of the character?

Typing Tutor

by Roy Groth

Wish you were a better typist, but don't want to take (or pay for) a class? Teach yourself to type with the aid of your microcomputer. With **Typing Tutor** you will be quizzed and graded, but



you set the pace at which you learn. **Typing Tutor** is a set of programs that lets you become as good a typist as you wish, allowing you to advance from one level to the next when you feel comfortable with your skills.

Let "hunt and peck" slip into the past, teach yourself speed and accuracy on the keyboard with **Typing Tutor**.

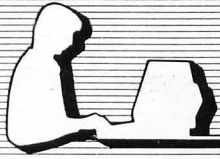
S-80 Mod I & III

16K Tape..... \$19.95

32K Disk..... \$20.95

SoftSideTM Selections

6 South Street Milford NH 03055



EPSON 100 F/T TYPE II PRINTER

by D.F. MACY

Epson has a new printer, the MX100 F/T Type II, an addition to its growing line of inexpensive peripherals. I received one today and proceeded to assemble it.

There are three ways to handle the assembly of any new gadget:

1. Open the box, remove the whatchamacallit and assemble (in case of difficulty read the instructions), or
2. Read the instructions and proceed to slip tab A into slot A, or
3. Read the instructions and hand the entire mess to your kid who will assemble it.

I made the mistake of reading the instructions which fell from the enclosed gray book entitled "MX-80 PRINTER TYPE II." (Right then I should have known something was wrong, but sometimes wisdom eludes me.) I removed a cable, a long box with numbers on it and two carefully padded bags. Next came the printer, embedded in styrofoam pillows. Removing the outer wrappings, I compared the parts with the instructions. Whoever wrote the assembly instructions never saw an MX100 printer. For those of you who have, or are planning to purchase an MX100, I offer this advice: Read pages 1-11 of the manual

and apply common sense.

Epson, on the sheet accompanying the booklet, has printed, "MX-100 is a sister model of MX-80 F/T TYPE II. It has greater print width than MX-80 F/T TYPE II to.....There's no other difference between the two models." Horseapples!

The Epson 100 is 22 inches wide and 20 inches deep, not including the separator and bail. The separator snaps onto the tractor assembly, not into the printer case. The paper bail is in three pieces which assemble into the rear of the case. Shipping screws are as shown. The printer cover slides from the printer in upright position without wrestling with the case. To lift the upper case, loosen five screws from the top and lift up. The roller knob need not be removed. The cartridge ribbon is a large scale version of the MX-80. Be careful with it. As of this writing, Epson does not have a replacement ribbon at any cost.

Operation of the Epson 100 F/T TYPE II is exactly the same as the MX 80 Friction/Tractor Image Dot Matrix Printer. I even used the accessory RS-232C plug-in interface (\$150) without difficulty with my Heath H-8. The

tractor feed is easily removable for individual sheet, word processing applications. The MX-100 will print 136 characters per line or 233 in the condensed mode on fanfold or sheet paper. It will accept paper from four to fifteen and one-half inches wide. Paper guides have been improved and the paper no longer "hangs up." If you intend to use the printer in the Bit Image mode (video pictures), plan for 816X8-1632X8 dots per line. I cannot imagine readers of **SoftSide** owning equipment for converting analog TV signals to digital for CRT viewing. Nor can I picture any with a program to convert these signals to eventual composite printout. (We have, at **Softside**, one individual who is capable of preparing such a program, but Rich would need the equipment.) However, if there are any of you out there with the equipment and ingenuity to program your personal computer, this is the printer you will need.

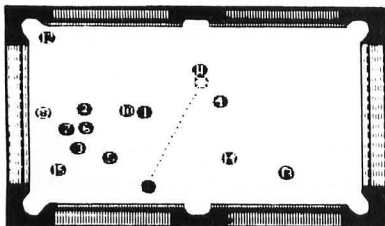
I have seen better printers than the MX-100. However, for the money (\$819 from TSE/Hardside), you can't beat it. If you are seriously considering a large format printer, I recommend you try the Epson MX-100 F/T TYPE II.



Pool 1.5 for the Apple II™

By

INNOVATIVE DESIGN SOFTWARE, INC.



The first and only HIRES color graphics pool simulation for the Apple II or Apple II Plus.

Requires: 48K Apple II with Disk II and Paddles ~~\$34.95~~ \$29.95

- Real-time color HIRES animation
- 256 directions for aiming
- 4 popular games
 - Eight Ball
 - Straight Pool
 - Rotation
 - Nine Ball
- Instant replay for any shot
- Special "slow-motion" control
- Pool 1.5 supports the best HIRES animation on the Apple today.





THE COMPUTER AND THE HOT DOG

Editor's note: With "The Computer and the Hot Dog," we welcome Dean Macy to the SoftSide staff. We hope you will enjoy his informational and witty style.

by Dean F.H. Macy

What a marvelous age in which we live.

When I first entered the computer field, a personal computer was any computer which could fit into a single room. It's name was IBM 1401, and it was the first computer with Random Access Memory. Prior to RAM, memory was entered and removed in sequential form.

How does SAM (Serial Access Memory) compare with RAM (Random Access Memory)? Let's say you put a hot dog on a bun, cover it with ketchup, mustard, relish, perhaps a pickle. You place the whole mess in a baggie and put it in temporary storage, the refrigerator. Several days later you remember it's there, and remove it for lunch (or whatever you do with an old hot dog). Using SAM you find exactly what you entered; a hot dog on a bun, covered with ketchup, mustard, relish, and perhaps a pickle. Using RAM you may find a pickle on a bun covered with ketchup and nothing else. During the time your hot dog was in storage, some ingredients were randomly moved to new locations. (Empty stomachs perhaps?) The missing items could have been replaced with grape juice and eggs. (On a bun?)

The 1401 memory core was a large frame filled with iron ferrite cores, each about the size of a ring. The frames were stacked in huge cabinets for easy storage. The larger companies had many such cabinets, each capable of housing nearly 5,000 memory cores. (WOW! Just think, a 5K memory, or with three cabinets 15K.)

Now all this may sound like a lesson in antiquity; but, in actuality, our present day RAMs do nothing more than the original cores so far as memory storage is concerned.

Many years ago it was found that when an electric current was applied to an iron core, the core became magnetized. When the core was magnetized, it was on; and when it was

demagnetized, it was off. (So, the cores had two states, on and off.) If we use the symbol "1" when the core was on and the symbol "0" when the core was off, we could use cores to count real numbers. Ergo, the binary system (base two).

In the early years of the computer era, as in these times, the system most used was the BCD (Binary Coded Decimal). The early computer used what today would be called a four bit memory processor; four bits (cores) were all that were needed to house one computer "word." Let's take a moment to analyze a four bit BCD word.

Remember we talked about "1" representing an on state and "0" an off state? The four bit core memory looked like this: 0 0 0 0. Each bit represented a number in the decimal system with the first, right hand bit equaling one, the next two, the next four, the next eight. To make this easier to understand, I'll put the decimal equivalents under the bits.

0 0 0 0
8 4 2 1

If the first and third position bits in the memory looked like 0 1 0 1, this equaled a decimal "5." If all bits were energized, the decimal equivalent was "15." If a decimal "16" is needed, we must go the next group of four bits. 0 0 0 1 0 0 0 0 = 16 decimal; and so it continues.

How do you add using a BCD system? Just remember that the largest number is "1" and it's quite simple. OK, I'll show you one, a hard one, and you do some easy ones to get the feel of what your computer goes through every time you give it an instruction.

0 0 1 1
+ 0 1 0 1

1 0 0 0

How'd we do that? Starting with the right hand figures, as in normal addition, and remembering that 1 is the highest number, think this through....1+1=0 and carry the 1. 1+0=1+the carried 1=0 and carry the 1. 0+1=1+the carried 1=0 and carry the 1. 0+0=0+the carried 1=1. Simple, huh?

No? OK, using the decimal equivalents of 8 4 2 1 on top, bits 2 and 1 on, equals 3. The lower 8 4 2 1, with

bits 4 and 1 on, equals 5. 5 + 3 = 8. 1 0 0 0 is the BCD equivalent of eight. Now you try some. (Get the hang of it and WOW your friends, kids, whomever.)

The year is now 1970. A new type of four bit RAM has been designed to replace the cores which were unable to withstand the great strain over continuous computer use. Instead of cores for memory storage, some genius (I use the word with reverence) discovered that by proper biasing, transistors could be made to turn on or off by application of positive or negative current.

They called these new-fangled devices "Flip-Flops." These circuits were etched by special process into a substrate we now call a "Chip." These chips evolved into IC's (Integrated Circuits) of great complexity. So complex are modern day IC's that what used to be housed in a large room now fits on your desk with room to spare. And what now covers part of your desk will soon fit in your pocket. (In a recent issue of BYTE Magazine the cover depicted a Disk-based computer system complete with CRT on a man's wrist. I shudder at the thought!)

The RAM's used for storage today are called 1K or 4K or 16K, the K signifying "Thousand." A 1K RAM contains 1024 sets of "Flip-Flops" which will hold 1024 bits in "On" or "Off" states.

In order not to confuse you, the reader, (Hello? Are you still out there?) I will confine all future references to a simple eight bit RAM, the most commonly used personal computer "word" length, commonly called 1 BYTE.

OK! We have a personal computer up and running with eight bits of RAM. (What does one do with eight bits of RAM?) The question is, "How does information get into the RAM?" There are several methods of transmitting positive and negative pulses to the flip-flops in the RAM; the more common types mentioned below.

"WARNING-DO NOT FOLD, STAPLE OR MUTILATE." Sound familiar? Our government still uses punch cards as checks (i.e. your Income Tax refund). Each card is a complete file containing information using

continued on page 87



THE NATIONAL COMPUTER SHOWS

HAVE WE GOT A PROGRAM FOR YOU IN '81 & '82

Attend the biggest public computer shows in the country. Each show has 100,000 square feet of display space featuring over 50 Million Dollars worth of software and hardware for business, industry, government, education, home and personal use.

You'll see computers costing \$150 to \$250,000 including mini and micro computers, software, graphics, data and word processing equipment, telecommunications, office machines, electronic typewriters, peripheral equipment, supplies and computer services.

All the major names are there including; IBM, Wang, DEC, Xerox, Burroughs, Data General, Qantel, Nixdorf, NEC, Radio Shack, Heathkit, Apple, RCA, Vector Graphic, and Commodore Pet. Plus, computerized video games, robots, computer art, electronic gadgetry, and computer music to entertain, enthrall and educate kids, spouses and people who don't know a program from a memory disk.

Don't miss the Coming Of The New Computers—Show Up For The Show that mixes business with pleasure. Admission is \$5 for adults and \$2 for children under 12 when accompanied by an adult.

Ticket Information

Send \$5 per person with the name of the show you will attend to National Computer Shows, 824 Boylston Street, Chestnut Hill, Mass. 02167. Tel. 617 739 2000. Tickets can also be purchased at the show.

THE MID-WEST COMPUTER SHOW

CHICAGO
McCormick Place
SCHOESSLING HALL
23rd & THE LAKE
THURS-SUN
SEPT 10-13, 1981
11AM TO 7PM WEEKDAYS
11AM TO 6PM WEEKENDS

THE MID-ATLANTIC COMPUTER SHOW

WASHINGTON, DC
DC Armory/Starplex
2001 E. CAPITAL ST. SE
(E CAP. ST. EXIT OFF I 295
-KENILWORTH FRWY)
ACROSS FROM RFK
STADIUM
THURS-SUN
SEPT 24-27, 1981
11AM TO 7PM WEEKDAYS
11AM TO 6PM WEEKENDS

THE NORTHEAST COMPUTER SHOW

BOSTON
Hynes Auditorium
PRUDENTIAL CENTER
THURS-SUN
OCT 15-18, 1981
11AM TO 7PM WEEKDAYS
11AM TO 6PM WEEKENDS

THE SOUTHEAST COMPUTER SHOW

ATLANTA
Atlanta Civic Center
395 PIEDMONT AVE NE AT
RALPH MCGILL BLVD
THURS-SUN
OCT 29-NOV 1, 1981
11AM TO 7PM WEEKDAYS
11AM TO 6PM WEEKENDS

THE SOUTHERN CALIFORNIA COMPUTER SHOW

LOS ANGELES
LA Convention Center
1201 SOUTH FIGUEROA
THURS-SUN
MAY 6-9, 1982
11AM TO 7PM WEEKDAYS
11AM TO 6PM WEEKENDS

tiny, square holes. A single row deciphers into one BYTE (eight bits) of information. Cards are read by light beams and photocells. If light strikes a photocell, a bit is turned on. No light and the bit is off. A similar format is punched tape, very popular with Western Union, both simple to use and abuse.

After generations of punched cards and tapes, users began searching for less costly formats. (Did you think some squirrel sat around pasting little squares back in the holes?)

Enter the TTY (TELETYPE-WRITER)! Let's briefly examine the ASR-33 by Teletype Corp.. This device offered a seven level ASCII character keyboard, a typing unit for hard copy, and a paper reader/punch. TTY's communicated with each other using messages on pre-punched tape. (Operators punched keyboard entry messages into the paper tape for later, high speed transmissions.) The TTY was interfaced with a computer using a serial code of eleven elements per character, in which "0" and "1" were called "Space" and "Mark." The first element was a "Start" bit followed by eight bits of ASCII code, with the final two bits used as "Stop" bits. The TTY transmits and receives at the outstanding speed of ten characters per second (110 BAUD). (Every once in a while, in a tiny town in the back of beyond, a LOL still bangs away on an old ASR-33, keeping in touch with reality.)

We've come a long way since the TTY, RTTY, storage drum and hard-core disk to the future frontier, the LASAR format. Now, in 1981, we are still using the magnetic format; reel to reel, cassette, stringy floppy and disk.

Whether you LOAD/DUMP on cassette or disk, I will use tape format to explain how data is sent to your RAM. Tapes are "loaded" using continuous recording techniques, where serial groups of eight bits form "characters" or "words." Groups of "words" (two to several thousand) form "blocks" or "records," and groups of "records" (two to several hundred) constitute "files." "Records" are separated by "inter-record gaps," .5 to 1 inch long. It's at "gaps" that your tape unit stops (When it stops at all). A typical cassette tape can record 700,000 characters depending upon density (bits per inch/ baud rate).

A tape "block" format includes a "start" character which indicates the beginning of a "block," "record-length" characters which notify the computer of the actual number of "bytes" in the "record," "load ad-

dress" characters which provide the starting address of the "load," and a "checksum" character which contains the sum of all "bbytes" in the "record." As the "record" is loaded into the computer, the computer counts all character bits and compares the result with the "checksum." Unless the two are equal, the load is aborted, and a "CHECKSUM ERROR" appears on the CRT.

Characters on magnetic tape are not "marks" or "spaces," or "1's" and "0's." Bits or characters are implanted using "frequency modulation." (What?!) Ever hear of an FM radio? Gotcha!! Digital signals are converted to one of two frequencies corresponding to the "0" and "1" values of the data. (Bits on tape systems are musical, somewhat, and if you carefully play your cassette tape on a player you will hear "musical data.")

Keeping the foregoing in mind, we'll load a tape into our eight bit personal computer. First, a look at the "record." Our "record" contains a "start bit" followed by a "record length bit." (The record length bit = 8.) Since our tiny computer always loads at the same address, there is no "address bit;" and at the end, a "checksum bit."

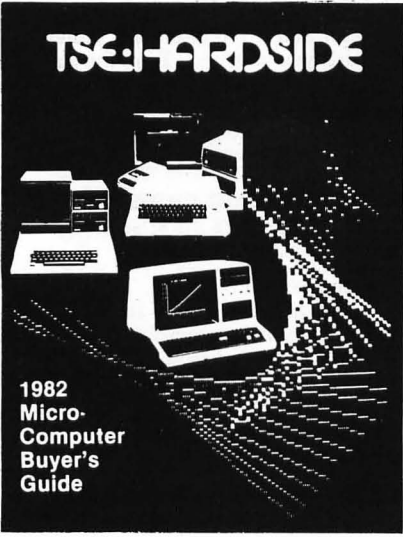
OK, load the tape. The computer

decodes the "music" into "1's" and "0's." These are usually loaded into a buffer which holds the bits until the microprocessor decides where to put them. Since our memory consists of an eight bit RAM, the bits are loaded in sequence.

Let's explore our RAM. The FM bits were coded to read, 00100011 (35 decimal). If you add, using the keyboard, (Can an eight bit computer have a keyboard?) a "BCD 8" to your RAM, your memory storage would read 00101011. (Can you decode to decimal? Remember each bit represents a decimal number; the eight bits corresponding to /128/64/32/16/8/4/2/1/.) OK, we're finished and ready to dump the contents of our updated RAM back onto tape. The computer codes the "1's" and "0's" to "musical" tones and a new record is created.

I have attempted to simplify the extremely complex workings of the LOAD/DUMP cycles of a personal computer. I realize I have omitted much, but to have included information concerning the many chip functions, I might have confused my readers. If you wish to further understand the wheeling-dealings of a computer system, drop me a line, and I'll try to comply.

THE TSE-HARDSIDE 1982 MICRO-COMPUTER BUYER'S GUIDE




TSE-HARDSIDE

1982
Micro-
Computer
Buyer's
Guide

is now available! We've included such valuable information as print samples from each of the printers we carry, feature-by-feature comparisons of Micro-Computer systems in an easy-to-read table format, an informative article on Micros, and pages and pages of complete product descriptions.

We're making this valuable reference available for only \$2.95 (refundable on your next purchase from TSE-HARDSIDE.) Charge customers are welcome to call our toll-free number: 1-800-258-1790 (in NH call 673-5144) THE TSE-HARDSIDE 1982 MICRO-COMPUTER BUYER'S GUIDE will soon arrive at your address via first class mail.

Send to:



TSE-HARDSIDE

Dept. C, 14 South Street
Milford, NH 03055

SS10

Yes! Send me the TSE-HARDSIDE 1982 Micro-Computer Buyer's Guide.

I've enclosed \$2.95 Please send only your FREE Price List

Charge to my credit card MasterCard Visa

Card No. _____

Interbank No. _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____

State _____ Zip Code _____



TYPE-'N-TALK™ IS T.N.T.

The exciting text-to-speech synthesizer that has every computer talking.

- **Unlimited vocabulary**
- **Built-in text-to-speech algorithm**
- **70 to 100 bits-per-second speech synthesizer**

Type-'N-Talk™, an important technological advance from Votrax, enables your computer to talk to you simply and clearly — with an unlimited vocabulary. You can enjoy the many features of Type-'N-Talk™, the new text-to-speech synthesizer.

You operate Type-'N-Talk™ by simply typing English text and a talk command. Your typewritten words are automatically translated into electronic speech by the system's microprocessor-based text-to-speech algorithm.

The endless uses of speech synthesis.

Type-'N-Talk™ adds a whole new world of speaking roles to your computer. You can program verbal reminders to prompt you through a complex routine and make your computer announce events. In teaching, the computer with Type-'N-Talk™ can actually tell students when they're right or wrong — even praise a correct answer. And of course, Type-'N-Talk™ is great fun for computer games. Your games come to life with spoken threats of danger, reminders, and praise. Now all computers can speak. Make yours one of the first.

Text-to-speech is easy.

English text is automatically translated into electronically synthesized speech with Type-'N-Talk™ ASCII code from your computer's keyboard is fed to Type-'N-Talk™ through an RS 232C interface to generate synthesized speech. Just enter English text and hear the verbal

response (electronic speech) through your audio loud speaker. For example: simply type the ASCII characters representing "h-e-l-l-o" to generate the spoken word "hello."

TYPE-'N-TALK™ has its own memory.

Type-'N-Talk™ has its own built-in microprocessor and a 750 character buffer to hold the words you've typed. Even the smallest computer can execute programs and speak simultaneously. Type-'N-Talk™ doesn't have to use your host computer's memory, or tie it up with time-consuming text translation.

Data switching capability allows for ONLINE usage.

Place Type-'N-Talk™ between a computer or modem and a terminal. Type-'N-Talk™ can speak all data sent to the terminal while online with a computer. Information randomly accessed from a data base can be verbalized. Using the Type-'N-Talk™ data switching capability, the unit can be "de-selected" while data is sent to the terminal and vice-versa — permitting speech and visual data to be independently sent on a single data channel.

Selectable features make interfacing versatile.

Type-'N-Talk™ can be interfaced in several ways using special control characters. Connect it directly to a computer's serial interface. Then a terminal, line printer, or additional Type-'N-Talk™ units can be connected to the first Type-'N-Talk™ eliminating the need for additional RS-232C ports on your computer.

Using unit assignment codes, multiple Type-'N-Talk™ units can be daisy-chained. Unit addressing codes allow independent control of Type-'N-Talk™ units and your printer.

TYPE-'N-TALK™ comes with:

- Text-to-speech algorithm
- A one-watt audio amplifier
- SC-01 speech synthesizer chip (data rate: 70 to 100 bits per second)
- 750 character buffer
- Data switching capability
- Selectable data modes for versatile interfacing
- Baud rate (75-9600)
- Data echo of ASCII characters
- Phoneme access modes
- RS 232C interface
- Complete programming and installation instructions

The Votrax Type-'N-Talk™ is one of the easiest-to-program speech synthesizers on the market. It uses the least amount of memory and it gives you the most flexible vocabulary available anywhere.

Order now. Toll free.

TERMS: Prices and specifications are subject to change. TSE HARDSIDE accepts VISA & MASTERCARD. Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). TSE HARDSIDE pays all shipping charges (within the 48 states) on all PREPAID orders over \$100.00. On all orders under \$100 a \$2.50 handling charge must be added. COD orders accepted (orders over \$250 require 25% deposit), there is a \$5.00 handling charge. UPS Blue Label and Air Freight available at extra cost.

#09-274001H **\$369.00**



TSE HARDSIDE

6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

BUGS, WORMS,



and other undesirables



There are a few bugs in the program QUEST 1 published in the August 1981 issue. The following are corrected lines to be typed in place of those appearing in print.

S-80 Version:

```
836 IF RC=0 THEN INPUT"IS (S)HE A DWAR
F";A$:IF LEFT$(A$,1)="Y" THEN RC=2
```

Also, here is an improved arrow firing routine sent in by Joe Sewell of Melbourne, Florida. It does not succumb to the occasional bugs found in the S-80 original:

```
15700 X6=X5*2-1:Y6=Y5*3:X7=WX*2-1:Y7=W
Y*3
```

```
15701 IFX6=X7THENSL=SGN(Y7-Y6):X8=X7+1
:Y9=X6+1ELSESLP=(Y6-Y7)/(X6-X7):IFX6>X
7THENX8=X6:X9=X7+4ELSEIFX7>X6THENX8=X6
+4:X9=X7
```

15702 Delete this line.

```
15705 IFX6=X7THENIFY7<Y6THENY7=Y7+3:Y6
=Y6-2ELSEIFY7>Y6THENY7=Y7-2:Y6=Y6+3
```

```
15706 Y2=Y7:IFABS(WY-Y5)=1THENY2=Y2+S6
N(SLP)*3
```

15707 Y8=Y6:Y9=Y7:Y=Y8

15708 Delete this line.

```
15710 FORX=X8TOX9+.1STEPSGN(X7-X6)
```

```
15720 SET(X,Y):Y=Y+SLP:IFX9<>X8THENNEX
TXELSEIFY<>Y2THENNEXTX
```

```
15750 Y=Y8:FORX=X8TOX9+.1STEPSGN(X7-X6
):RESET(X,Y):Y=Y+SLP:IFX9<>X8THENNEXTX
ELSEIFY<>Y2THENNEXTX
```



Apple Version:

```
836 IF RC=0 THEN INPUT"IS (S)HE A DWAR
F? ";A$:IF LEFT$(A$,1)="Y" THEN RC=2
945 P1= INT((P1*2+A)/3): IF (P1<=A) TH
EN 941
```

Here is a solution to the "OUT OF MEMORY" error, deftly worked out by Tom Pollard of Dunstable, Mass.

```
15110 Y5=Y5-1: IF Y5<2 THEN RM=R2(RM,1
): POP: GOTO 1000
```

```
15210 Y5=Y5+1: IF Y5>22 THEN RM=R2(RM,
2): POP: GOTO 1000
```

```
15310 X5=X5+2: IF X5>39 THEN RM=R2(RM,
3): POP: GOTO 1000
```

```
15410 X5=X5-2: IF X5<1 THEN RM=R2(RM,4
): POP: GOTO 1000
```

```
15620 X=RND(1): IF RC=1 THEN X=X+.1
```

Atari Version:

```
836 IF RC=0 THEN ? "IS (S)HE A DWARF "
:;INPUT A$:IF A$(1,1)="Y" THEN RC=2
874 GOTO 884
```

Also, here is a change suggested by the Atari author:

```
5025 POKE106,PEEK(106)+5:GRAPHICS0:RUN
20065 ? :? "Would you like to try agai
n":? "as a *NEW* character ";:INPUT A$
:IF A$(1,1)="Y" THEN 5025
```

This eliminates the need for hitting (SYSTEM RESET) prior to every RUN under normal condition.

The machine language loader program for Divide and Conquer published in the June 1981 issue was not the final version. To correct this, replace line 50010 with:

```
50010
POKE16553,255:POKE16561,128:CLEAR50
```

Then delete lines 50050 thru 50080. Note that the previous bug correction is still required.

OOOOPS! Even the BIG BOSS can make a small mistake. Due to the little known fact that the TRS-80 CLEAR statement is limited to a value of 32767 (negative numbers won't work), in the rare case of a person with a 48K cassette system, an ?OV ERROR will be generated in the program COMPUTESKETCH in the AUGUST 1981 issue. To remedy this, type in the following:

```
100 CLS:CLEAR50:IFMEM#.8>32767THENCLEA
R32767ELSECLEAR(MEM#.8)
102 N=FRE(A$)/900:DEFINTD-K:DIMP$(12,N
)
```

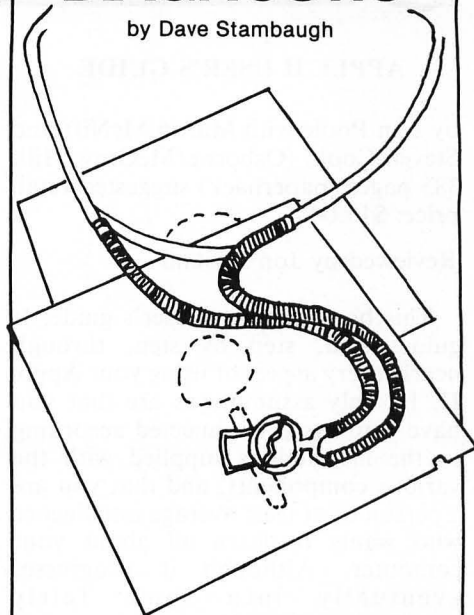
In the listing of the Atari version of "Battlefield" (August, 1981), a couple of characters were dropped from the end of line 11000. The line should read as follows:

```
11000 IF CP(CC,CR)<>PL THEN N6=0
```



FLOPPY DISK DIAGNOSTIC

by Dave Stambaugh



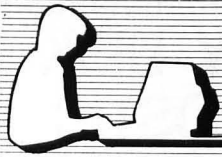
Now includes memory diagnostic at the same price

The best and most complete diagnostic you can buy to verify disk drive reliability and find problems. Displays 19 error messages and cross references them to 14 possible causes. Continuous test option for exhaustive testing keeps statistical record of all errors found.

- 35 or 40 track in same program
- Tests controller functions and status bits
- Tests drive speed and allows adjustment
- Tests switches and mechanical components
- Verifies data transfer
- Tests drive seek function
- Sector and byte write and read tests using all possible patterns
- 16 to 48K, 1 to 4 disk drives
- Tests cross cylinder interference
- Tests drive-to-drive compatibility

Supplied on diskette with manual for only \$24.95. \$22.50 (Model I only)

**SoftSide
Selections**
6 South Street Milford NH 03055



APPLE II USER'S GUIDE

by Lon Poole with Martin McNiff and Steven Cook. (Osborne/McGraw-Hill, 385 pages, paperback) suggested retail price: \$15.00

Reviewed by Jon Voskuil

This book is truly a user's guide: It guides you, step by step, through nearly every aspect of using your Apple II. Its only assumptions are that you have your system connected according to the instructions supplied with the various components, and that you are a person of at least average intelligence who wants to learn all about your computer. Although it progresses eventually into some fairly sophisticated discussions and programming techniques, a total neophyte should have little problem picking up the book and working his way through it. It would be a good book to bring home with your Apple the day you buy it, or to go back and buy when you get bored with running canned programs and want to get into the "hard" stuff.

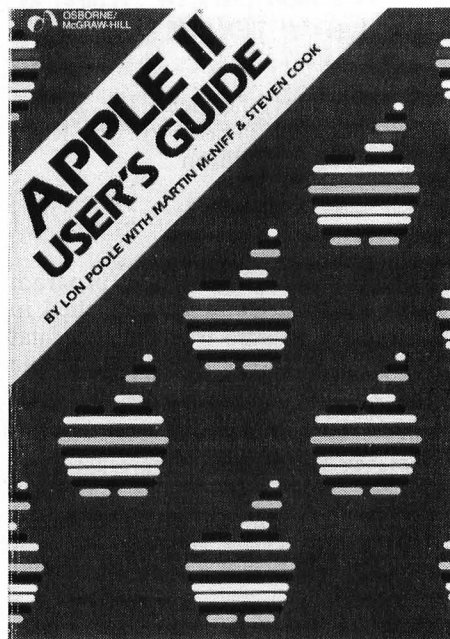
The chapter titles give the overall organization of the book:

1. Presenting the Apple II
2. How to Operate the Apple II
3. Programming in BASIC
4. Advanced BASIC programming
5. The Disk II
6. Graphics and Sound
7. Machine Language Monitor
8. Compendium of BASIC Statements and Functions

Following these are twelve appendices which provide a lot of good quick-reference material on functions, editing, errors, PEEKs and POKEs, memory usage, and the like. A fairly comprehensive (seven-page) index fills out the rest of the 385 pages.

The overall quality of the writing is very good. Explanations are clear and logically ordered. A number of black-and-white photographs grace the first two chapters and chapter five, picturing the system components and interconnections. There are almost no diagrams or sketches in the rest of the book, but I didn't feel that any were needed. The authors make good use of charts and tables throughout, and include many sample program listings and printouts to assist in understand-

ing the text. The typeface is easy to read, and the pages have a pleasing appearance. Only two errors caught my eye: one on page 40 where "dollar sign (\$)" should be replaced by "question mark (?)" as the abbreviation for PRINT, and one on page 156 where a table of Applesoft commands should be headed "Deferred Mode Only" rather than "Immediate Mode Only."



The book does not talk down to the reader; neither does it lapse into jargon which would leave the uninitiated feeling confused and irritated. (The only irritation I experienced was with the use of "data" as a singular noun — alas, an all-but-universal phenomenon in recent writing.) While tackling new material one step at a time, the authors do not hesitate to delve into the finer points of a topic when they need to do so. They do not assume that the reader wants to "skip over the hard stuff," and they do a commendable job of presenting occasionally difficult material in an understandable way.

This is the only book I've seen that can stand alone as a complete introduction to the Apple II. Since it is system-specific, you don't have to sort through material which does not apply to your system (a plus for the beginner), and it covers all the Apple's graphics and sound capabilities. I can't help but recommend it.

POOL 1.5

from Innovative Design Software, Inc. Apple II or II Plus, 48K and disk drive Suggested retail price: \$34.95

Reviewed by Rich Bouchard

The documentation for Pool 1.5 calls it "...one of the finest HIRES animated graphics games ever written for the Apple II." Pool 1.5 lives up to the description. This simulation comes as close to the "real thing" as possible. The program is written entirely in machine language and uses full-color high-resolution graphics. The graphics and sound effects are extremely well done and remarkably life-like. Great attention is given to making the table look realistic: The bumpers are green; the cue ball is white; and the playing balls are either red, or blue with a white stripe. The balls can also be displayed as white with black numerals to play other games.

The realism extends into the play of the game as well. When you scratch, you have the option of returning any pocketed balls, and the cue ball is repositioned by the other player, as in real pool. If you try to position it illegally, a warning tone will be heard, although the ball can still be shot from the illegal position. The interaction of the balls themselves is very realistic. Striking pockets off center or striking a corner pocket parallel to a bumper can cause a ball to bounce back out. Replaying the shot at a slightly slower speed may allow the ball to be sunk. This program also does a very good job of simulating the physics of pool. For example, when a moving ball with no English on it strikes a stationary ball, all the energy is transferred to the stationary ball. If the cue ball strikes several balls in a line that are touching, only the end ball will be affected.

Pool 1.5 allows up to four players, and plays four variations of pool: Eight Ball, Rotation, Straight Pool and Nine Ball. Straight Pool is the simplest game; turns rotate among the players until all the balls are sunk. As long as a player sinks a ball, he may continue to shoot; otherwise, play is passed to the next player. After all the balls have been sunk, the player who pocketed the most balls wins the game.

In Eight Ball — basically a two-player game — each player tries to pocket his group of “high balls” (9-ball through 15-ball) or “low balls” (1-ball through 7-ball). The first ball sunk determines which group of balls belongs to which player. After a player has legally pocketed all the balls in his group, he then shoots at the eight ball. While shooting at the eight ball, a player loses if he scratches. As in real Eight Ball, the player also loses if he sinks the eight ball before all of his other balls have been sunk.

In Rotation, the cue ball must strike the lowest numbered ball on the table before it hits any others. If this is done, the player gets points for every ball pocketed on the shot. The number of points is equal to the number of the ball. A miss, or failure to strike the object ball, passes play to the next player. If balls are pocketed illegally, the program provides a chance to return them to the table.

Nine Ball, the last variation of pool supported by this program, is similar to Rotation, except that only nine balls are on the table. One point is scored each time the nine ball is sunk, and the nine ball is returned to the table. All the other rules of Rotation apply. This game is very challenging, in that it forces the player to try some interesting combination shots to score well.

You aim your shot using a paddle control. A “ghost ball” is displayed showing the trajectory that your shot will take, as well as how the cue ball will strike the object ball. You can set the force with which you will hit the cue ball; the lowest setting is a nudge, while the highest is the type of hit that often results in balls on the floor in real pool. You can also add English to your shot: You can hit the cue ball high or low, to the left or right (or a combination of those two), or dead center. With practice, you can use combinations of aim, force and English to become proficient in sinking shots that would make Minnesota Fats envious.

The game has a number of useful commands which can be used during a game. You can set the speed with which the balls move, as well as the amount of friction (or rate at which the balls decelerate). The lowest friction setting, according to the program documentation, is “...like playing on a table with no cloth.” Breaking, with the lowest friction and the highest speed settings, rapidly causes many of the balls to fly around the table, and will usually sink most, if not all, of the balls before motion finally comes to a halt. There is also a feature that repositions the balls to their locations before

the last shot, and allows a replay of that shot.

One must look very hard to find anything wrong with this program. The one thing I noticed it lacked was the ability to position the playing balls to set up trick shots and the like. But this is really quite minor. With all the options and variations allowed, there is more than enough variety to keep one from hungering for more features.

This program is one of the most enjoyable I have ever seen or played, and is a game that will long remain one of your favorites. The suggested retail price of \$34.95 may seem a little steep, but the game is well worth it. Beware, this game may produce a hard-to-resist “one more game” compulsion.

FIRST AUDITION FOR “ORCHESTRA 80”

from Software Affair, 858 Rubis Drive, Sannyvale, CA, 94087. S-80 hardware/software package. Suggested retail price: \$80.

Reviewed by Robb Murray

Editor's Note: At press time, SoftSide discovered an expanded version of the “Orchestra 80” had been made available. Watch for a review of “Orchestra 85” in our November issue.

Overture

“ORCHESTRA 80” allows the user to both code and play first-rate music. Its appearance on the market is part of what is unquestionably a major advance in the democratization of composing.

The precision and reliability of computer processing have never been more justifiably or nobly invoked than in the performance of music by the computer. Until recently, however, computer-produced music has only been possible through the use of very expensive equipment and complex programming routines.

Here's how it works: First, the user loads a compiler program into main storage. This step sets up the system both for playing pre-existing code and for receiving new or changing old code. If the user wishes to play music, he runs a file of coded symbols through the processor. As this happens, output from the job is produced in the form of electrical signals. The signals flow through a digital/analog converter attached to the side port on the expansion interface (or, on the 16K S-80, to the port in back of the keyboard).

The attachable converter changes the raw electrical signals it receives into

electrical frequencies which, in turn, are amplified and drive a speaker (both the amp and speaker system are user-supplied). If the user wants to code music, he enters edit mode and begins to write a new file of code, or change a file already present. When finished, he compiles and stores the result, ready for playing.

What does “Orchestra 80” sound like? Well, for openers, the system does produce good musical sounds; surprisingly good ones in fact. Timing is precise, and tone color is pleasing. “Orchestra 80” will play up to four simultaneous voices over a six-octave range.

Fanfare

The value of “Orchestra 80” will depend on the user's motivation and musical background, but several uses for the system are apparent.

First, “Orchestra 80” allows you to play pre-coded music. Over a hundred musical renditions for “Orchestra 80” are already available, ranging from Bach to Stevie Wonder. Many bulletin boards offer local contributions; a board in Connecticut presented Christmas tunes a year ago.

Used in this way, “Orchestra 80” functions like a digital player piano. If this were all it could do, “Orchestra 80” would be a fabulous novelty item (player pianos are fun!); but it offers more.

Because the code that is played can actually be scanned and edited by the user, even the non-composer can make experimental changes to “canned” music. This feature makes it possible to play with coded music as though it were silly putty — bending, stretching, compressing, and molding it into a custom-made version of the musical work in hand. Just imagine a sixty-second Mahler's First Symphony, or a two-hour Minute Waltz. Learn “Orchestra 80” coding and imagine no more!

“Orchestra 80” enables you to transcribe standard music to computer-playable format. This process may offer untold rewards to the person who can't play an instrument (or can't play well), yet still longs to “play” favorite pieces for himself.

“Orchestra 80” can function as a working tool for the composer. It allows music to be tested, changed, and played at any speed. It is more than a compositional aid; it is also a mind-bending new performance medium which offers new freedoms and strengths to the composer. The limitations of traditional performance technique such as keyboard reach,

continued on next page

continued from previous page

vocal range, or ease of bowing are no longer at issue. If you can code it, the computer can play it; it's as simple as that.

To an admirable degree, "Orchestra 80" has set the standard of performance for its inevitable successors. Its music is always played nimbly, regardless of speed. When it comes to pace and precision, this system easily outperforms any keyboard musician. It never hesitates, stumbles, or tires.

As many as four voices may sound at the same time. Most of the previous music systems for the S-80 could play one or sometimes two layers of musical sound at once. Multiple-voice effects, such as Walter Carlos showcased in his popular "Switched-On Bach" albums of analog synthesis, had to be achieved with the aid of a tape recorder and multitracking or overdubbing. "Orchestra 80," however, plays up to four voices together, in real time. This represents a fundamental advance over previous systems, and its musical implications are obvious. With the new possibility of four-part counterpoint and harmony, most contemporary music, not to mention a large part of traditional music, has become suddenly and astonishingly accessible to the computer musician.

The coding scheme is simple and logical. I learned it in half an hour and, in person, could explain it to another user in ten minutes (it does help to learn it with a friend). To begin a piece, the user first writes a few lines of set-up code that specify the key, speed of performance and timbres of the four voice-lines to be used. Then the piece of music is coded, one measure at a time, and within each measure, one voice at a time.

Each note is given a time value and pitch value (which may include notation for accidentals). For example, "Q0#" means that a quarter note ("Q") is to be played at the pitch of C-sharp ("0" equals middle C; "0#", therefore, equals middle C, sharped). The time values of notes may range from tied whole notes to sixty-fourth notes (the latter may also serve as grace notes). Triplets, happily, may be coded, and at any speed.

When all measures have been coded, a four-character dummy measure signifies the conclusion of the piece or movement.

"Orchestra 80's" music is easy to test. For the composer, this feature is invaluable. Even before a piece is played, it is automatically screened by the compiler for certain common coding errors. Messages to the user show errors and where in the file they

occur. The user can then correct them and go on.

During listening sessions, playback need not commence at the beginning of a piece, but instead can start at any chosen point; thus a change made midway through a long piece can be tested immediately without having to suffer through a replay of all prior passages.

Similarly, playing may be interrupted at any point simply by touching the BREAK key. However, once stopped in midplay, a piece cannot continue automatically from that point unless a break-point has been previously set there.

Another valuable feature is the ability to slow playback to a crawl, allowing one to check the audio output against a musical score.

Tempos are easy to change. The user can speed or slow the tempos of entire pieces, or of sections, either by changing the tempo parameters in the set-up code or by pressing certain keys during performance. Thus, tempo accelerations, retardations, and "rubato" (combinations of which are used to convey interpretative feeling) may be either coded into the music itself, or created at will during playback. Tempo settings are accomplished with tempo-dedicated keys and a chart in the instruction manual, relating combinations of keys to set-up code.

Transposition from one key to another is automatic. It may be directed up or down, and applied to entire pieces or only sections of them. This is achieved by specifying, in the set-up code, the number of steps up or down needed to reach the pitch level at which the music is desired to sound. Major/minor transpositions are also possible by changing the coded key signature according to standard musical rules. Each of the four possible voices may be individually transposed. This allows easy coding, directly from the staff of music written in the special clefs (alto, tenor, etc.) for scorings of band or orchestral instruments.

"Repeats" save coding time and space, much as they do in traditionally notated music. Repeats of notes, phrases, passages, and entire pieces are readily codable.

Articulation is adjustable. Four special symbols paired with designated notes allow the user to separate consecutive notes to a greater or lesser degree. Normal note-separation is "legato" (a somewhat joined sound), but the special symbols provide different qualities of "staccato," in which notes are sounded with sharper attacks and quicker decays than usual.

One more nice feature of the system

is its price. Other music systems I have heard about cost from two to four hundred dollars. They give you more, but they aren't designed for the S-80.

Blues Tempo

Accompanying the nice features are some disappointments. First, the coding scheme doesn't resemble written music. There is usually no vertical alignment of notes that sound together. This makes it impossible to see harmonic relationships. Even simple mop-up work on one's compositions, such as checking for parallel fourths, fifths, and octaves, and scanning for similar (versus contrary) motion, is virtually impossible to do while working in computer code. Composing "at the tube" is out of the question.

The transcription of music similarly requires a painful intermediary step: writing out the code by hand before entering it into the machine.

A number of systems now on the market make use of a light pen that writes directly on a musical staff displayed on the tube. This feature gives these systems a vital advantage — almost without regard to what else they may or may not do. However, as far as I know, such an advanced graphics capability is not yet available for the S-80.

Dynamic (loud/soft) changes cannot be coded. They can only be imposed from the outside, by adjusting the volume level of the auxiliary speaker system. This means that no accenting of notes is possible with "Orchestra 80," nor are crescendos, decrescendos, or relatively loud and soft parts or passages. No doubt about it; that's bad.

Conspicuous overtone distortion occurs in the highest octave, though it is not unbearable. And, distributed over the remaining octaves, there is a faint, fuzzy quality that puts the unmistakable stamp of COMPUTER on these sounds.

The codable octave range is limited to four octaves. Even though "Orchestra 80" plays six octaves, one can actually code only within four. Transposition can expand the playing range, but, though the user may have access to all playable notes during an extended performance, only four octaves are available at any given moment.

Note separations in one part may distort the flow of other parts. True splitting may not be occurring from an engineering standpoint, but the effect of voice motion in an inner part (alto or tenor, for example) may cause a distracting break in the flow of

soprano and bass sounds.

Harmonic moments cannot be sustained during playback. When testing code, the user can slow performance to a very low speed, but can never quite freeze it. "Freezing" would sustain the notes being played together indefinitely, and would be useful for the careful scrutiny of coded harmonies. This problem is almost the only regard in which "Orchestra 80" is not an ideal system for testing musical code. A choir or orchestra, by contrast, frequently does hold a sustained note or chord during practice so that the close analysis of a particular musical moment can be made.

The system cannot be tuned. If a user wishes to code an accompaniment to a record album or to play a guitar along with the computer, it would be delightful to be able to tune the computer up or down a fraction of a step to bring it into accord with the other instrument. Alas, this is not a real option with "Orchestra 80."

As is true with so much system documentation, the accompanying user's manual for "Orchestra 80" is long (39 pages) and a little frightening at first glance. To make matters worse, it is organized from a computer-technical, not a musical, standpoint. The section that actually teaches how to code comes last — the worst place for it — and even then it is too long, chatty and familiar in style to be conveniently referenced later. More and better headings would begin to improve this section.

Back to Refrain

I am a Baroque music aficionado who likes to write music. My problem has always been getting my work played. I am a slow sight reader who can't do justice to good music on the keyboard, even to my own compositions. In the past I have gone so far as to try to recruit musicians from a chamber orchestra who would play numbers I have written, but with no luck.

Now I'm using "Orchestra 80," and finding musicians is no longer a major concern to me. The computer is the most precise and untiring of performers. It is true that once coded, digital computer music is "fossilized," i.e., neither computer nor user can subsequently add any feeling to it. But, unlike most musicians, the computer won't add any mistakes to it, either; and the computer never needs to practice. For me, that's a better-than-even trade-off, especially where contrapuntal music is concerned.

The first time I heard a Bach fugue

played on "Orchestra 80," I kept waiting instinctively for the music to bog down, or for a mistake of some kind. But, no matter how tightly the fugal mesh was woven, every note came in right on cue, sharp as a pinpoint. No doubt Bach himself would have appreciated the kind of relentless, unerring stream of musical sounds that "Orchestra 80" can produce. For my part, I'm delighted!

I have a number of friends who, knowing a little about music, tried writing it, and gave up. They liked the idea of composing and thought they could do it, but lack of keyboard ability always kept them from perfecting their musical ideas. "Orchestra 80" could have helped them. If you are such a person, it might help you.

Finale

I want to encourage amateur composers to take advantage of the peculiar structures and freedoms offered by "Orchestra 80." If you compose, or would like to, give the system a try. If the results are good, publicize them on one of the bulletin boards.

While not the same as totally new composition, I would also like to hear some good, creative examples of altered compositions, that is, arrangements made by changing the

coded versions of music composed by others. It would be especially nice to hear some good arrangements created by someone with no formal musical training. I'm curious to see what some grade-school student will produce, astounding everyone.

Standards are already being set by people such as Roy Niederhoffer, whose "Star Wars" is far and away the best thing on "Orchestra 80" yet released. There is also Bryan Eggers, whose transcriptions probe as well as anyone's the new potentialities offered by note-processing via microcomputer.

But it is now time for users to go beyond nearly literal transcriptions from the standard pop, classical, and novelty repertoires — and on to new, original compositions. Personally, I have been so excited about "Orchestra 80" that I have coded a half-hour recital of some of my own work. The results have been good enough to convince two Chicago radio stations to air the selections. Still greater things are possible!

Computer music has been around for years, but never as accessibly. Microcomputers are creating historic days in music. If you love music, own an S-80 and want to make a little history, consider buying "Orchestra 80."



At Last! Orchestra-80

A TRS-80™ MUSIC SYNTHESIS SYSTEM

WRITTEN BY JON BOKELMAN

Turns Any 16K Level II TRS-80 Into A High Quality Musical Instrument

The Software

A five part machine language program consisting of:

- 1 Digital synthesizer**—produces up to four simultaneous voices in a six-octave range.
- 2 Music language compiler**—a simple and easy to use language allows you to enter your favorite written music in any key or time signature.
- 3 Full screen editor**—a full function text editor with blinking cursor is provided for easy entering and modifying of music programs.
- 4 File manager**—provides the orderly storing and retrieval of named program files on tape or disk. You can even sequence several songs for automatic loading and playing.
- 5 Initialization**—this set-up routine allows you to alter the voices, select the standard four-voice synthesizer or a special high resolution, three-voice version and choose the standard (1.77 MHz) or the enhanced (2.66 MHz) clock rate.

The Hardware

A single 1½" by 2" PC board plugs into the expansion connector on the TRS-80 keyboard or the screen printer connector on the expansion interface. This board contains the electronics required to convert the computer output into a high fidelity audio signal. Just plug in the board and connect to the aux/tape/tuner input of any audio amplifier. No external power supply is required.

Includes:

- Tape and disk versions on cassette
- Completely assembled and tested PC board
- Detailed and complete instruction manual
- Sample music programs

Orchestra-80

~~\$80.00~~
\$69.95



™TRS-80 IS A TRADEMARK OF TANDY CORPORATION

We help you reach out to the world.

PERSONAL SOFTWARE INC.™

VISIDEX

The VisiDex™ program makes it a snap to remember anything, and everything, you don't want to forget.

VisiDex imposes no restrictions on the format or content of your data. So you can file away any kind of information you want to: names and addresses, important numbers and dates, ideas, things-to-do, lists, notes, report and memo highlights, stock, tax, and personal information.

And you can retrieve exactly what you want, instantly, because VisiDex "thinks" the way you do; giving you unlimited cross-referencing to help you find everything faster.

It's like having "total recall," not just for one piece of information, but also for every other associated piece of data you've filed away.

VisiDex gives you unprecedented flexibility in the way you can handle computerized information—no planning, no customizing, no formatting is necessary. It's like a big box of blank cards. You type in whatever you want on a "card." And whatever key reference words, phrases, numbers or dates you want and file it away. Because you decide how to cross-reference your "cards," VisiDex "remembers" the way you do. You just give it the key reference you "probably" filed it under (like the date, or a customer's name, for example).

Visdex
#47-204004D \$199.95

VisiDex will instantly give you back everything you've ever filed away that included that particular reference (like the notes from your last meeting with the customer, the customer's organization chart, the customer's personal likes, hobbies, birthday, ideas you intend to propose to the customer, ad infinitum!)

VisiDex will also print out its information as mailing labels, lists, memos, or other formats you specify, all in alphabetic or numeric order, or by date order. VisiDex will watch over your calendar, too, and alert you to special dates, appointments and occurrences.

Besides these unstructured filing capabilities, VisiDex also gives you the ability to structure the information you'd want filed consistently, creating a form with optional built-in keywords. That way you can have your secretary add or change information on these forms, and be assured that you get all the information filed correctly.

And, like other Personal Software programs, VisiDex lets you exchange data with other programs. Pass your notes and memos to VisiTerm, where you can transmit or receive them over a phone line connected to a co-worker's personal computer. That's "electronic mail" in action! Or incorporate portions of a VisiCalc analysis or estimate into a memo which you've filed away.

VisiDex is available for Apple II and II Plus personal computers.

If you don't ever want to forget anything important ever again, remember to have your Personal Software dealer show you all that VisiDex can do.

The Desktop/PLAN II™ program gives you a fast and easy way to automate and organize all your financial analysis, budgeting, and business planning.

The program guides you step-by-step in easy stages, from the original layout of your budget or plan through final computerized calculation and print-out of a boardroom-quality report. It can produce its own line charts and bar graphs, too—for greater visual clarity.

Complex and frequently used financial calculations—even multi-step operations using chains of tests and formulas—become easier, quicker to handle. And Desktop/PLAN II can handle very large financial "models," too. This makes it especially useful for consolidation across departmental budgets and combining the results of identical models into an overall company budget with ease and simplicity.

If you currently use timeshared financial modeling programs, you'll find Desktop/PLAN II familiar and easy to use—and no more monthly timeshare bills! If you've never done financial modeling, Desktop/PLAN II can introduce you to an easier and more productive way of financial planning and budgeting than the laborious manual methods of the past.

Desktop/Plan II
#47-204008D \$199.95

DESKTOP/PLAN II

You can produce a high-quality, professionally organized print-out of your plan or model with Desktop/PLAN II. The format is much like an accountant's columnar analysis pad—it can be up to 18 columns wide or up to 300 rows deep. Pagination, headers which carry your model's title, and other report features are handled automatically by the program.

And Desktop/PLAN II can receive information from VisiCalc—an interchange capability giving you greater combined usefulness.

Desktop/PLAN II offers exceptionally powerful planning system features in a package which operates at your desk and at your convenience.

Desktop/PLAN II is available for Apple II and II Plus personal computers.

Ask your Personal Software dealer to demonstrate its planning power for your business and financial plans.

VISITERM

With the VisiTerm™

program, you and your personal computer can communicate with larger computers, with other personal computers, and send and receive disk information over the phone.

You'll be able to tap into time-sharing services, newswires, data bases, stock reports and more. You'll be able to access your company's own dp computer, if there is one, for data entry, proprietary software, and internal communications.

Visi Term
#47-204003D \$149.95

VISICALC

The VisiCalc™

program is the most widely-used personal computing program in the world today!

That's because VisiCalc is the easiest to use, most versatile way for you to "run the numbers" when evaluating any financial or business decision.

Visicalc
#47-207010D \$199.95

VISILOT

The VisiPlot™

program adds a striking new visual dimension to all your financial analysis, forecasting, budgeting and business planning.

VisiPlot takes all those rows and columns of hard-to-grasp numbers and automatically turns them into easy-to-comprehend charts and graphs, right before your eyes.

Visi Plot
#47-204001D \$179.95

VISITREND/ VISILOT

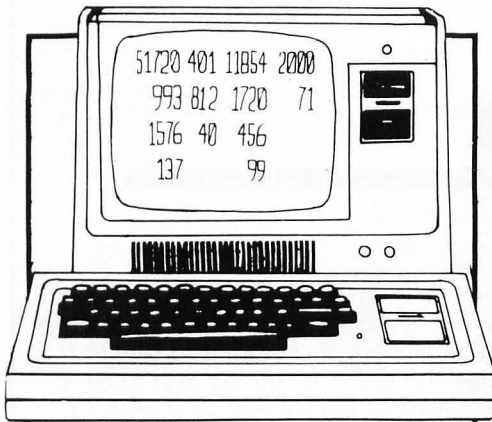
If you want sophisticated trend forecasting and statistical analysis—and you know how hard it is to create them by hand—you'll want to get your hands on the VisiTrend/VisiPlot™ program!

VisiTrend/VisiPlot responds to the demanding needs of the sophisticated manager, planner, investor, analyst, statistician, consultant. You'll be able to accomplish more because you decide on your time series analysis, and VisiTrend/VisiPlot performs it for you. So you can spend your time examining various alternatives, not figuring them out.

Visi Trend/Visi Plot
#47-204002D \$259.95

TERMS: Prices and specifications are subject to change. **TSE HARDSIDE** accepts VISA & MASTERCARD, Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). **TSE HARDSIDE pays all shipping charges (within the 48 states) on all PREPAID orders over \$100.00.** On all orders under \$100 a \$2.50 handling charge must be added. COD orders accepted (orders over \$250 require 25% deposit), there is a \$5.00 handling charge. UPS Blue Label and Air Freight available at extra cost.


6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE
1-800-258-1790



COLUMN CALCULATOR 4.1

by David T. Gray

Column Calculator is a "word processor for numbers," a number processor designed to be used like a calculator. But it can handle large blocks of information as if handling one number at a time. The work space can be thought of as a large matrix with rows and columns much like an accountant's spreadsheet. Data can be easily entered into columns; and the columns can then be moved around. Columns can be overlaid from an existing data file on disk. One column can be added, subtracted, multiplied, divided, or raised to a power of another and the results put in another column. Columns can be compared to one another. Columns can be totalled, or set with a constant, and any column can be sorted, carrying the rest of the columns with it. A predefined function can be defined, thereby preprogramming the worksheet.

The statistical section provides analysis of the data. The analysis includes simple statistics, linear regression, simple correlation, histogram and the T-test.

The information can be printed out on the line-printer in a compressed format at any stage in the development of a data base. Thus, it can be used as a finished report or as a copy of the worksheet to permit the filling in of additional data for later entry into the data base. The data base can be saved on disk and recalled at a later date for modification or for generating a report. Any column in a file on disk can be referenced and added to the current worksheet.

All user communication with **Column Calculator** uses FLASH, the line input/editor routine. This enables the user to not only key in instructions, but to edit errors or data as well.

S-80, 32K disk \$39.95

**SoftSide
Selections**
6 South Street Milford NH 03055

HARDCOPY



**OKIDATA
Microline 83**

Direct from TSE HARDSIDE

- NEC 5520 SpinWriter (09-223036H) \$3195.00
- NEC 5510 SpinWriter (09-223035H) \$2795.00
- NEC 5530 SpinWriter (09-223037H) \$2795.00
- CENTRONICS 737 (09-256002H) \$749.00
- OKIDATA Microline-80 (09-258001H) \$439.00
- OKIDATA Microline-82 (09-258003H) \$579.00
- OKIDATA Microline-83 (09-258004H) \$879.00
- EPSON MX-80 (09-223038H) \$519.00
- EPSON MX-80FT (09-223039H) \$619.00
- EPSON MX-100FT (09-223041H) \$829.00
- NEC Tractor-Feed Option (09-223033H) \$249.00
- C.ITOH 25cps Daisey (09-259001H) \$595.00
- C.ITOH 45cps Daisey (09-259002H) \$1995.00
- C.ITOH Tractor Option (09-259003H) \$189.00
- BDT Sheet-Feeder (NEC only) Option (09-223034H) \$1495.00
- Microline-80 or 82 Tractor-feed Option (09-250002H) \$59.00
- Centronics Zip-Pack Ribbons (3-pack) (15-267001S) \$15.95
- SpinWriter Multi-Strike Ribbons (3) (15-223040S) \$19.95

- Microline Printer Ribbon (3-pack) (15-258006S) \$15.95
- EPSON Printer Ribbons (2-pack) (15-223042S) \$29.95
- RS MOD-I & III Printer (02-208015H) Cable (26-1401) \$29.00
- RS MOD-I & III Printer (40-pin) Cable (02-208016H) \$29.00
- APPLE Parallel Int. & Cable (36-pin) (07-223016H) \$100.00
- APPLE Parallel Int. & Cable (40-pin) (07-208014H) \$100.00
- APPLE Asynchronous RS-232C Int (07-249002H) \$159.00
- ATARI-Macrotronics Print (36-pin) Int(06-254001H) \$69.95
- ATARI-Macrotronics Print (40-pin) Int(06-254002H) \$69.95

TERMS: Prices and specifications are subject to change. TSE HARDSIDE accepts VISA & MASTERCARD. Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). TSE HARDSIDE pays all shipping charges (within the 48 states) on all PREPAID orders over \$100.00. On all orders under \$100 a \$2.50 handling charge must be added. COD orders accepted (orders over \$250 require 25% deposit), there is a \$5.00 handling charge. UPS Blue Label and Air Freight available at extra cost.

TSE HARDSIDE
14 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

SoftSide™

BACK ISSUES

SOFTSIDES 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 articles and the systems they're for. Each issue costs \$3.00 for the magazine only. Those issues marked with an asterisk are also available on cassette for \$9.95 or disk for \$14.95. September 1981 is the first enhanced Disk Version and costs \$19.95.

SoftSide S-80 Edition:

October 1978 — Not Available
November 1978 — "End Zone"
December 1978 — "Santa Paravia En Fiumaccio"
*January 1979 — "Round the Horn"
*February 1979 — "Form 1040"
*March 1979 — "Personal Finance"
*April 1979 — "Safari"
*May 1979 — "Dog Star Adventure"
*June 1979 — "Atlantic Balloon Crossing"
*July 1979 — "All Star Baseball"
*August 1979 — "Meltdown"
September 1979 — Not Available
*October 1979 — "Westward 1847"
November 1979 — Not Available
*December 1979 — "Oil Baron"
*January 1980 — "Moving Maze"
February 1980 — Not Available
*March 1980 — "Broadway"
April 1980 — Not Available
*May 1980 — "Star Trek"
*June 1980 — "Micro Millionaire"
*July 1980 — Adventure Issue

PROG 80 (S-80 only)

March 1979 — Not Available
May 1979 — "Clock Routines"
July 1979 — "Histogram and BASIC Statistics"
September 1979 — "DOS-How to stop lost data errors"
December 1979 — "Computer Telephone Dialing"
February 1980 — "Hexmem"
April 1980 — "Redefining Level II Keyboard"
June 1980 — "Z-80 Disassembler"
August 1980 — "Varlist"—A program to list variables

SoftSide: Apple Edition

*January 1980 — "Dog Star Adventure"
*February 1980 — "Connection"
*March 1980 — "Treasure Hunt"
*April 1980 — "Jig-Saw"
*May 1980 — "Invaders"
June 1980 — Not Available
*July 1980 — "Pork Barrel"

SoftSide: Apple, Atari and S-80 Combined Edition

*August 1980 — "Caribbean Cruising" — Apple
 "Master's Golf" — Atari
 "Sailplane" — S-80
September 1980 — Not Available
*October 1980 — "Developing Data Base II" — All Three
 "Moonlanding" — Apple
 "World Series" — Atari
 "Earth-Port II" — S-80

*November 1980 — "Developing Data Base III" — All Three
 "Collision" — Apple
 "Trench" — Atari
 "Kriegspiel" — S-80

*December 1980 — "Developing Data Base IV" — All three
 "Baseball" — Apple
 "Speedello" — Atari
 "Kidnapped" — S-80

*January 1981 — "Developing Data Base V" — All three
 "Convoy" — Apple and S-80
 "Angle Cannon" — Atari
 "Ship Destroyer" — S-80

*February 1981 — "Developing Data Base VI" — All three
 "Miner" — All three
 "Mini-Golf" — Atari and S-80
 "Long Distance" — S-80

*March 1981 — "Developing Data Base VII" — All three
 "Strategy Strike" — Apple and S-80
 "Flags" — Atari
 "Volcano" — S-80

*April 1981 — "Battle At Sea" — Apple
 "Convoy" — Atari
 "Dominoes" — S-80

*May 1981 — "Galaxia" — Apple
 "Dodge" — Atari
 "Orienteering At Jacques Coulee" — S-80

*June 1981 — "Old Glory" — All three
 "Word-Search Puzzle Generator" — All three
 "Anallist" — S-80

*July 1981 — "Chemistry Drill" — All three
 "Kidnapped" — Apple and Atari
 "Magic Paper Calculator" — S-80

*August 1981 — "Quest I" — All three
 "Battlefield" — All three
 "Compu-Sketch" — S-80

*September 1981 — "Flip-It" — All three
 "Word Challenge" — All three
 "Exterminate" — S-80
(Enhanced Disk Version — \$19.95)

Use the bind-in card in this issue to order or send a list of the back issues you'd like, with payment of \$3.00 per magazine to:

SoftSide Publications
515 Abbott Drive
Broomall, PA 19008

For the magazine/media combination, send \$9.95 per cassette and magazine, or \$14.95 per disk and magazine (\$19.95 for September 1981) to:

SoftSide Publications
6 South Street
Milford, NH 03055

First it will awe you...

Step beyond TRS-80™ entertainment software as you know it with **diversions thru Envyrn™**, the new bimonthly series. Boot up for the first time and you'll be astonished by what you find:

Experience three scales of graphics — take in a long, wide view of your **Envyrment™**, then step closer and still closer to zero in on detail.

Travel at your chosen speed — when your life depends on escaping, sprint. When you want to take a closer look at your **Envyrment™**, stalk.

Diversions™ intelligence will challenge you. Only the **Envyrment™** knows the location of a black hole, whether a bridge will collapse if you're carrying too much weight, or which walls are really secret entryways. It's up to you to uncover the hidden mysteries.

Wait until you discover the magnitude of the area open to your exploration! **diversions™** maps and diagrams are larger than any you've experienced. As you approach a boundary, your **Envyrment™** may load yet another realm from the disk, much like the unfolding of a road map.

After you've examined the features that make **diversions™** a revolution in computer entertainment, then discover the thrill of experiencing! Imagine yourself commander of the starship Enterprise, a paladin on a medieval quest, or the pilot of a mission flying over the Himalayas.

Subscribe for a full year and receive six **Envyrment™** modules with **diversions thru Envyrn™** the magazine — not a manual, but a full-sized handbook with history, hints and information that will intensify your experience of **Envyrment™**.

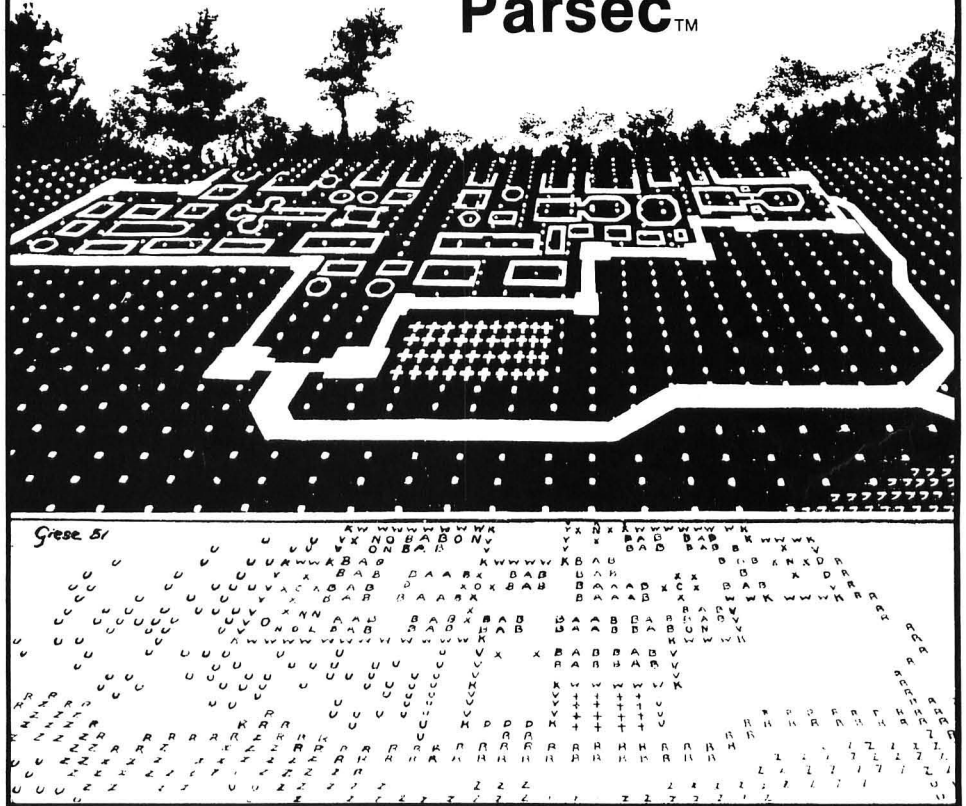
Send \$60 for one year (six modules). You must have a 48K TRS-80™ Model I or III with disk. **diversions™ thru Envyrn™**, Apple edition, will be available late winter, 1982. You may want to take advantage of our special introductory offer: \$20 for the first **Envyrment™**, **Parsec™**, what Star Trek has always tried to be.

Entertainment software will never be the same!

diversions™

thru Envyrn

Premiere Issue Parsec™



Send to:

diversions™

thru Envyrn

6 South Street Milford, NH 03055

- One year subscription (6 modules), I enclose \$60.00
 Sample **Envyrment™** Only **Parsec™** I enclose \$20.00
 MasterCard Visa

Card# _____
Interbank# _____ Exp. Date _____

Name _____

Address _____

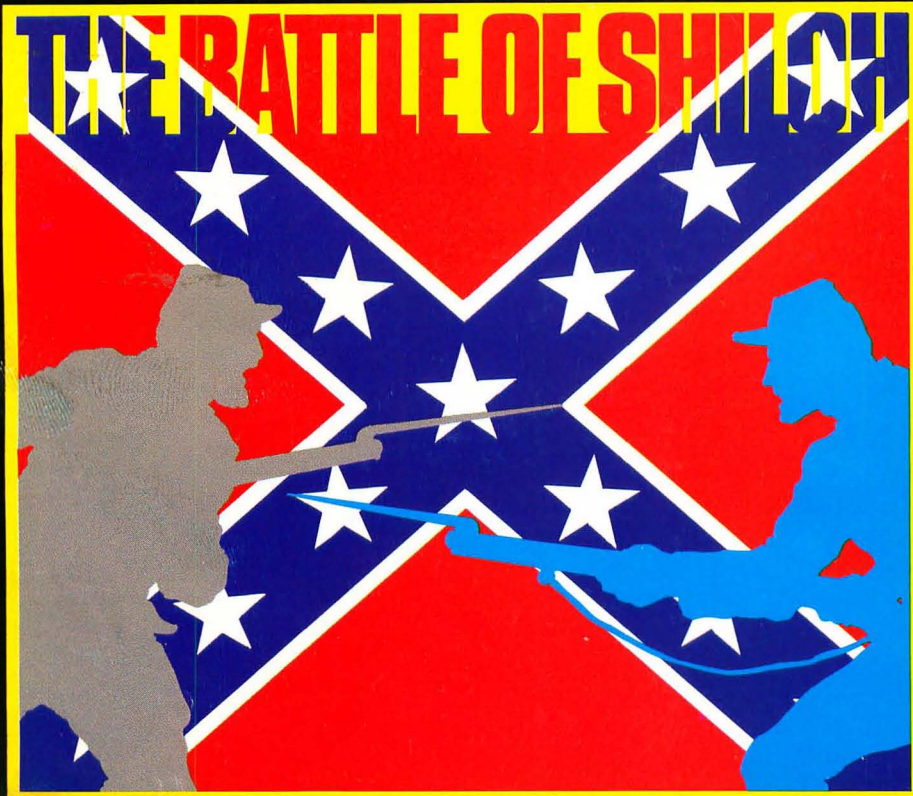
City _____

State _____ Zip _____

I own a 48K TRS-80™ Model I Model III

TWO NEW GAMES FROM SSI FOR THE

APPLE® AND THE TRS-80®!



THE BATTLE OF SHILOH: A brigade-level simulation of the first grand battle of the Civil War, pitting the Confederate Army against Grant's troops and Union gunboats.



TIGERS IN THE SNOW: Ghostlike Nazi Tiger tanks and infantry sweep across the dark, frozen forests of the Ardennes against a surprised U.S. force in this division/regiment-level simulation of Hitler's last desperate attack.

We know it hasn't been easy for you TRS-80® owners to see so many great made-for-Apple-only games from SSI pass you by. But then, it hasn't been easy for us to design games for a 16K cassette format good enough to meet our critical standards.

After all, we've got a reputation to protect, a reputation in strategy gaming for unsurpassed sophistication, innovation, realism, and playability.

Well, our designers have been hard at work, and we've not only met but surmounted the challenge. We're delighted to announce two historical wargames — deserving of the SSI label — for both the Apple® and the TRS-80® (16K cassette for the TRS-80 Model I and III; 48K disc for Apple II with Applesoft ROM card).

Combining our extensive wargame-design experience and superior programming techniques, we've given a fresh new look and feel to these favorite classic battles.

At \$24.95 each for TRS-80 cassette and \$39.95 each for Apple disc, these are extraordinary games at quite an ordinary price.

So head on down to your local store and check them out today!

VISA and M/C holders can order by calling 800-227-1617, ext. 335 (toll free). In California, call 800-772-3545, ext. 335.

To order by mail, send your check to: Strategic Simulations Inc., Dept. SS-1, 465 Fairchild Drive, Suite 108, Mountain View, California 94043.

All our games carry a 14-day money-back guarantee.



STRATEGIC SIMULATIONS INC

As part of our demanding standards of excellence, we use MAXELL floppy discs.

Apple is a registered trademark of Apple Computer Inc.

TRS-80 is a registered trademark of the Tandy Corporation.