

AUGUST 1981

SoftSide™

THREE DOLLARS

VOLUME IV • Your BASIC Software Magazine • NUMBER ELEVEN

Battlefield
Shark
Compu-Sketch
S-80 Video Reverse
VARPTR Used
and much,
much more!



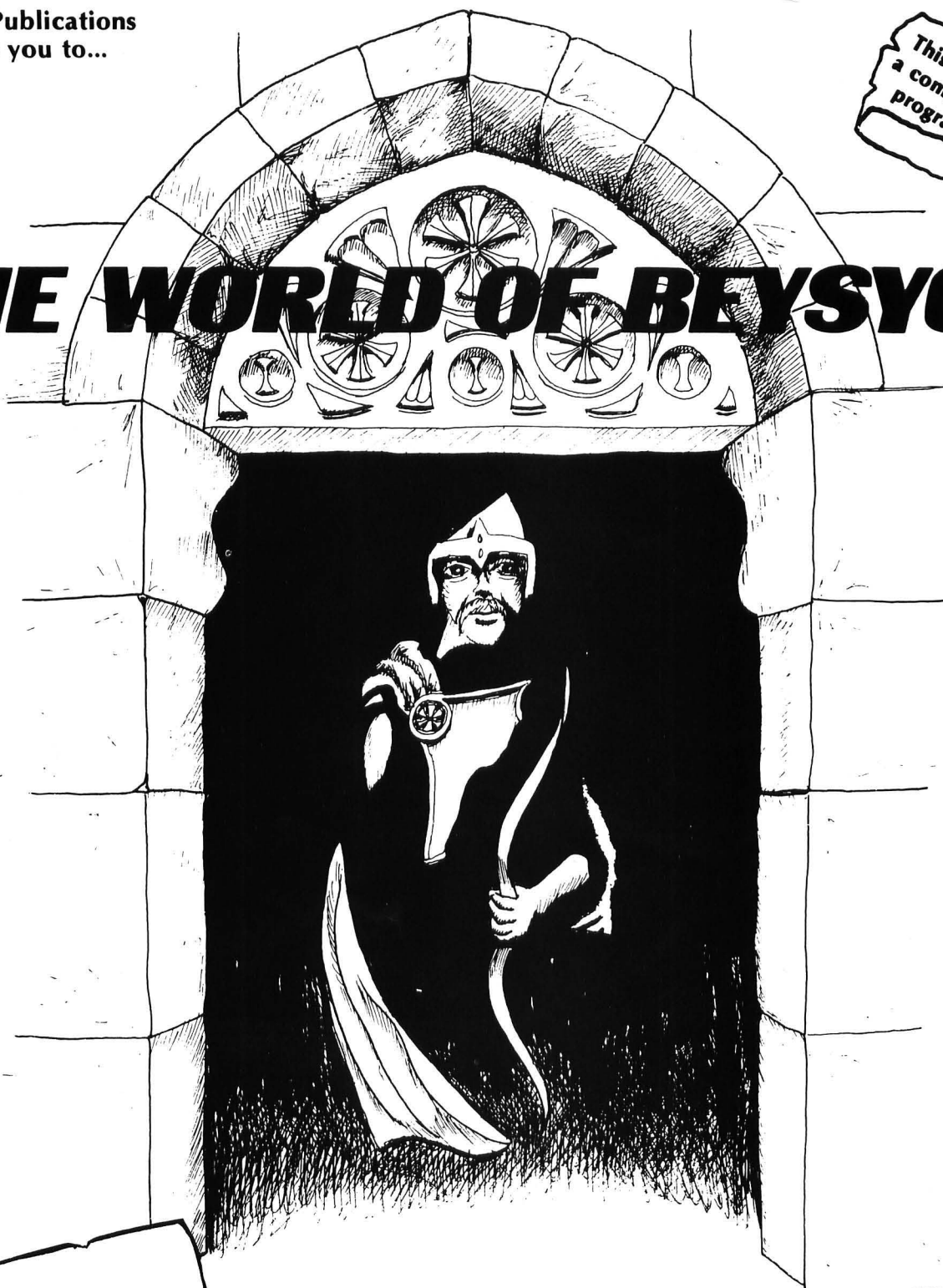
Quest 1

William A. Giese '81

SoftSide Publications
welcomes you to...

This is not
a computer
program

THE WORLD OF BEYSYCX



"...but all legends agree that, somewhere in the wilderness of the Loulanix Mountains, the severed arm of St. Elsinore rests in a temple carved out from the living rock...waiting for those pure in heart and mighty enough to return it to the Elven people."

A New Generation of Role Playing Adventures

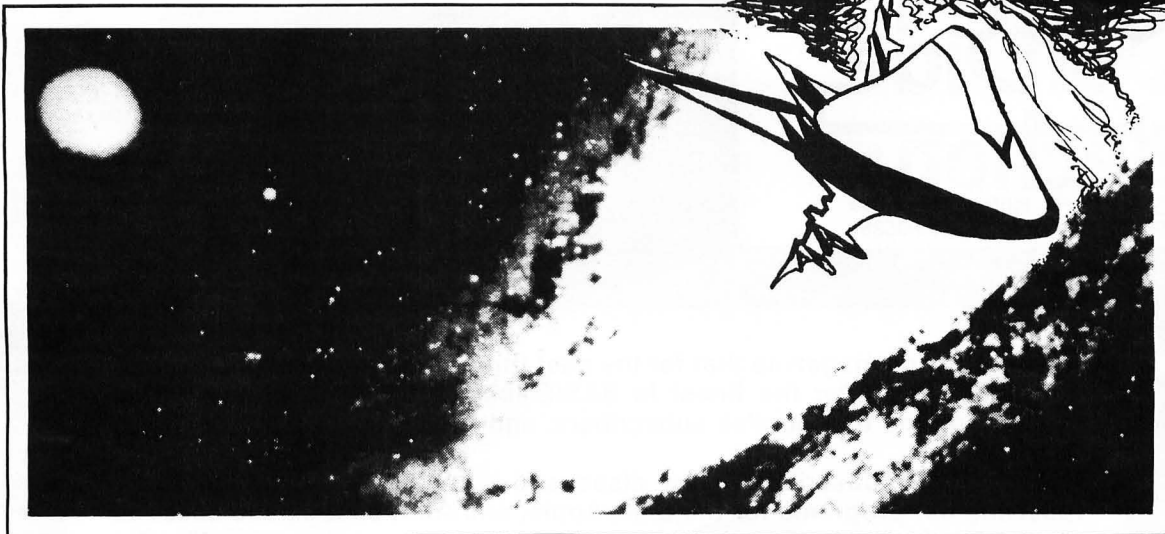
With all new monsters and treasures never before encountered, **The Quest for the Arm of St. Elsinore** is the premiere module in the **World of Beysycx** series of role playing adventures.

The **World of Beysycx** series was created for use with the popular Advanced Dungeons & Dragons (tm) gaming system, and is designed as the ultimate in DM playing aids. Each module comes with graphic cardboard map pieces and two fully illustrated booklets detailing the scenario and showing every monster and every treasure.

To order, send \$6.95 (plus \$1.00 shipping) to:
SoftSide Publications
6 South Street
Milford, NH 03055

Advanced Dungeons & Dragons is a registered trademark of TSR Hobbies.

**Hot pursuit
through space
and the
vortices
of time!**



RAINWARE PRESENTS...

Time Lord

The fallen Time Lord, who presumptuously calls himself The Master, is at large. The elders of Waldrom have supplied you with the hyperspace-worthy vessel TARDIS, and commissioned you to eliminate the evil "Master". Your resources include clones who will fight for you, the formidable CRASER weapons of the TARDIS, and magic weapons such as Fusion Grenades and Borelian Matrix Crystals.

Travelling 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?

Based on Dr. Who of PBS fame.
Apple Integer Basic,
Disk, 48K ... \$29.95



**SoftSide
Selections**

6 South Street Milford NH 03055
For Orders Only 603-673-0585

SoftSide-DV: The Magazine that Addresses Both of You...

**SoftSide
Selections**
6 South Street Milford NH 03055
For Orders Only 603-673-0585

SoftSide, the magazine that for the past three years has brought you and your microcomputer the finest in BASIC language software, now offers more programs to S-80 disk subscribers, and the software isn't limited to BASIC!

Over 2,000 subscribers have dispensed with the hours of typing by receiving our programs for the S-80, Apple, and Atari on disk and cassette along with the printed magazine, but convenience is only the beginning for the S-80 SoftSide-DV. Now if you subscribe to SoftSide S-80 DV you will receive all the programs in the magazine itself and a host of others, including Machine Language and hybrid (multiple language) programs that we could never print on paper. We will offer complete programs of every conceivable type and ongoing data bases. We will venture into totally modified languages, new versions of BASIC authored specifically for your S-80, and I-String programs with new data bases that will allow you to enjoy puzzles, games and adventures without having to type them in and thus have them half-solved before you ever start!

If you don't believe we can deliver what we're promising, try us out on a one-time basis: a full disk of programs for \$19.95. Of course if you subscribe, the cost will be substantially less: Until August 31 we will offer a full year's subscription to SoftSide-DV for a mere \$99.95, that's a \$8.33 per disk, and you get the magazine to boot!

For your convenience, we offer an installment plan for Mastercard and Visa card holders: You pay \$32.50 per quarter (price includes \$5.00 charge for the service). For orders outside the U.S.A. please add \$18. If you currently receive the magazine, we will credit the remainder of your subscription to a new SoftSide-DV order.

All subscribers to S-80 SoftSide-DV will receive a disk full of programs as well as SoftSide magazine, guaranteeing you not only the best software for your microcomputer, but the least expensive as well. It's our way of inviting you to step into the future.

Atari and Apple DV slated to be enhanced soon.



TABLE OF CONTENTS

ARTICLES

- 72 **Computer Graphics** Reflective Symmetry Joan Truckenbrod
76 **Video Reverse Modification** S-80 Do it yourself Ed Umlor
80 **Lemonade or Champagne** The last round Will Hagenbuch
84 **VARPTR Used** Would you buy a used VARPTR from this man? John T. Phillipp, M.D.
92 **Reviews** APPL-L-ISP, 3-D Computer Graphics Package Mark A. Ohlund, Alan J. Zett

S-80, APPLE, AND ATARI PROGRAMS

- 24 **Quest 1** Look out for the Wraiths Brian Reynolds, Rich Bouchard, Alan J. Zett
42 **Battlefield** Haul out the Howitzers Joe Humphrey, Jon Voskuil

S-80 PROGRAM

- 52 **Compu-Sketch** Graphics galore Roger W. Robitaille, Sr.

APPLE PROGRAM

- 59 **Shark** Just when you thought it was safe to go back to the keyboard Mark Cross

ATARI PROGRAM

- 64 **Dairy Farming** Cream of the crop Scott Tapley, David H. Simmons

DEPARTMENTS

- 4 **Editorial** Dave Albert
6 **Input** From our readers
7 **Outgoing Mail** Dave Albert
11 **Calendar of Events** Editors
13 **About This Issue** Drs. Munchkin
14 **Say Yoho** I. Alexis Adams
16 **My Side of the Page** Lance Micklus
18 **Machine Head** Spyder Webb
20 **The Sensuous Programmer** "J"
22 **Bugs, Worms, and Other Undesirables** Editors
88 **What's New** Ed Umlor
89 **Hardware Corner** Ed Umlor

Cover illustration by Bill Giese

STAFF

PUBLISHER:

Roger Robitaille Sr.

EDITORIAL DEPARTMENT

Scott Adams
Dave Albert
Rich Bouchard
Mary Locke
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Jon Voskuil
Alan Zett

PRODUCTION DEPARTMENT

Lynda Fedas
Damian Henriques
Nancy Lapointe
Tom Stanton

STAFF

Patricia Acampora
Lester Anderson
Brian Berkebile
Diana Bishop, **Subscriptions**
Kathleen Boucher
Suzanne Breton
Philip Brown
Tara Butler, **Dealer Information**
Jeff Carroll
Donna Cookingham
Pam Demmons
Mary Edwards
Mark Eric, **Atari Submissions**
Mary George
William F. Gollan
Kathleen Hannon
Donna Jean
Bette Keenan
Bea Kimball
Randal Kottwitz
Karen Lawrence
Kathy Maloof
Jean Matthews
Doris Miller
Clem Morey, **Apple Submissions**
Carol Roane
David Robitaille
Elizabeth Robitaille, **Personnel Administrator**
Cindy Schalk
Ken Sicard, **S-80 Submissions**
Joanne Tracy
Anmar William
Nancy Wood
Ed Umlor
Gary Young
Cynthia Zawacki

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
6 South Street
Milford, New Hampshire 03055



Editorial

by Dave Albert

Computers and Fantasy Role-Playing Games

We've all seen them before — fantasy game programs, the ones that come in the packages decorated with fiery demons, hideous serpents, and the occasional scantily clad maiden. They promise breathtaking excitement and at least two or three light years of enjoyable playing time... While the claims of the advertisements, both direct and subliminal, may be somewhat exaggerated, the games themselves are really a lot of fun. We've seen quite a few of them come through our doors, but none quite worth printing until this month's cover feature came to us. We decided to forego depicting ladies too impoverished to afford decent clothing, and we don't promise you'll have to be pried away from the keyboard with a crowbar once you play this game, but we've wanted to run a program like this one for some time.

"Quest" by Brian Reynolds of Huntsville, Alabama, is a solid computer adaptation of a Fantasy Role-Playing game (FRP). Those of you familiar with the Automated Simulations/Epyx series of FRPs ("Hellfire Warrior", "Temple of Apshai", et al.) will recognize the format of this one. But there is an added feature to this program: magic use. For those of you who are wondering just what this drivel you're reading is about, let me backtrack a little.


FRPs represent the new kid on the block in American gaming. The games are very open-ended and the setting for each is distinct, but the most common type is based on heroic fantasy literature. The name usually used to describe that literary genre is Swords & Sorcery, which rather succinctly expresses the two prime elements of the imaginary world in which the games are played. Swords, bows, spears, and axes are the most technologically sophisticated weapons, while spell casting and enchanted items are commonplace. So are monsters, demons, oversized animals, and carnivorous insects, not to mention undead creatures and their ilk.

To play, one assumes a character of heroic proportions, like a fighter or a magician of some sort. There are more

ways to create such a character than you can shake a halberd at, but it is usually done randomly, with dice. Each game system has a set of characteristics (Strength, Dexterity, Charisma, etc.) which defines the character's abilities. Then the character equips himself with whatever is available which pertains to his profession (Fighter, Magician, Thief, Cutpurse, Patchpurse, whatever) and sets out looking for treasure and adventure, not necessarily in that order. Treasure can be in simple coin of the realm, or as complex as arcane scrolls containing some powerful enchantment legible only to those who speak a language forgotten centuries before. And so forth. Each game has a combat system should the hardy adventurer encounter one of the aforementioned nasties, and some sort of magic system that permits the casting of spells.

Such games, born in this country in the early '70s, are naturals for computer applications. Adventure (with a capital "A"), the truest computer game, has some obvious ties to FRPs, both thematically and in the parameters of permitted activities. The primary difference between FRPs and their silicon chip counterparts is that the computer versions are usually solo games, while the traditional FRP is a multi-player game run by a referee of some sort. But then the computer FRPs have matters such as memory space to deal with, a limitation foreign to the pencil and paper variety.

Anyway, one of the things we liked about "Quest" is the inclusion of magic use by the player, a factor absent in most of the computer FRPs that we've seen around. And the fact that, while written in BASIC, it still works better than some of its more famous Machine Language counterparts didn't bother us a great deal either. Of course it is still only a single player game, and it is finite due to the limitations of the computer, but it is a lot of fun to play.

So take a look at a new kind of computer game in **SoftSide** and let us know how it sits with you. If you like it, see if you can do a better one. The possibilities are limited only by your own imagination. 



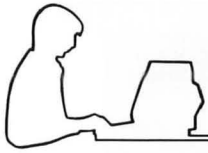
LORDS OF KARMA

The player finds himself in a mythical, magical city to perform as many deeds of kindness and bravery as possible. The fun is in the gradual deciphering of the many secrets while exploring verdant forests, twisting trails, rugged mountains, and labyrinthine caverns. . . and avoiding lurking monsters. Almost no rules to learn; you tell the computer what you want by typing simple English sentences on the keyboard. The computer responds with many surprises in store. Solitaire only.

S-80 Cassette, 48K.....	\$20.00
Apple II Cassette, 32K.....	\$20.00
PET 32K	\$20.00

**SoftSide
selections**
6 South Street Millard NH 03055
For Orders Only 603-873-0585

AVALON HILL



Dear **SoftSide**,

Re the letter on page 8 of your May, 1981, issue which says "... At least when you start reading, you don't read something that has nothing to do with your system."

In that same issue were two programs for the S-80 which I as an Apple owner would have completely missed had I followed Miss Taylor's suggestions. Both "Dairy Farming" and "Orienteering" require very little modification to go into the Apple and are well worth reading even though they were not for my system.

I believe most articles usually have an iota of knowledge contained in them which can be applied in some way to your computer education and maybe to your system, and shouldn't be written off because they weren't for your system.

I now have two nice (and long) programs I otherwise wouldn't have had.

A.W. Blackburn
Fort Worth, TX

Dear **SoftSide**,

I'd like to thank you for a marvelous first issue upon my return to the flock. I had been one of those who dropped out after the switch to the multiple machines listing. I decided to look at a friend's copy and saw what had gone on and found the transition better than the original versions, so I rushed out my subscription check and got your wonderful June issue.

The point of this letter is after looking over the listing of "Catacombs Of The Phantoms", I figured it could be converted easily for the S-80. I was right and here is what must be done:

First I changed three of the variables, WORM becomes WRM, Go becomes GO

(that's a zero, not an O in the new variable), and STONE becomes STN. Next, I made the following line modifications:

```
111 GA=0:GD=0
361 - eliminate the SETCOLOR command
590 - eliminate the SETCOLOR command
2065 IF LEFT$(A$,1)="N" THEN140
1350 IF LEFT$(A$,1)="N" THEN300
4000 REM
Delete 4010 to 4106 (except a print line for Tom's byline)
3000 PRINT" PRESS ANY KEY TO CONTINUE";
3010 A$=INKEY$:IFA$="
""THEN3010
3015 GOTO110
```

These modifications will allow the program to work. Now for a slight improvement:

```
316 IFR=200THENPRINT
"NOT HERE.":GOTO220
```

This statement prevents a BAD SUBSCRIPT error from occurring.

I'd like to make a few quick suggestions before I sign off here — possible additions to the magazine. How about a monthly column that will show hackers like me the modifications needed to run programs presented in previous issues in their machine's language (like I've done with the above)? You could post a small payment for readers who contribute (say \$10.00 per published mod.) and thus make your publication more versatile. Second, how about some tutorial articles on graphics for the S-80 and other machines? I find myself often lacking in this knowledge because I shift machines (including Heath, CP/M, PET, DEC, XITAN) so often at my job and everybody could use this kind of help.

Joseph Teller
Waltham, MA

Dear **SoftSide**,

Let me begin by saying how much I enjoy your magazine. I would, however, like to see more tutorial articles included, which brings me to one of two reasons I've decided to write. In past issues I have read many letters that complain about not having enough programs or too many programs published for a particular computer. On that I would like to say that whenever a program is not written for your machine, study the program anyway; there are many techniques that can be learned and applied to your machine. Another solution to this problem would be for **SoftSide** to run a series of articles on the differences between S-80, Apple, and Atari BASIC so that programs could be easily converted from one machine to another.

Now for the second reason I've decided to write: One of the most important aspects in computer work is the ability to sort a list of items alphabetically, such as a mailing list. Unfortunately, most of the very fast (Machine Language) sorting programs that have appeared in magazines only sort on one field, when, for instance, a mailing list may have five or more fields (name, address, city, state, and zip) per entry. So now you've got a program that sorts one field and you need one that sorts five — throw the program away and sort the list by hand, right? Wrong! Let's say, for instance, that you have a list of 500 entries in the form Name, Address, City, State, Zip and you need the list sorted by a particular field, say by name. There are then six steps that need to be followed to sort a multi-field list with a single field sorting program; they are:

1. Pick a character that does not appear in any of your entries, for instance "*".
2. Enter the information into the computer, either by hand or off tape or disk.
3. After each entry is entered, compact the fields into a single string with the field that needs sorting at the front of the string. When compacting the fields, separate the fields by the character chosen in Step 1. In this case the string would be in the form "Name*Address*City*State*Zip*". If the list needed sorting by zip code, the string would be in the form "Zip*Name*Address*City*State*". You now have one string that can be easily sorted by a one-field sorting program.
4. Sort the list.
5. Each string must now be separated into individual fields. This can be done with the mid-string or in-string functions.
6. As each string is separated into individual fields, store the separated string on disk, tape, or print the information out.

This method should work very well on STRING information, but numerical information may require additional work. I hope that this method alleviates some of the problems encountered when sorting multi-field lists when a single field sorting program is the only one available. I would appreciate hearing from those that try this method (problems solved or encountered).

Kevin D. Rich
Box 344
Perrysville, OH 44864

Dear **SoftSide**,

You folks seem to like taking it on the chin — I really enjoy reading the Input section, but sometimes it reads more like World War III. Frankly, I think **SoftSide** is great!! I subscribe to eight different micro-magazines, and most look forward to **SoftSide** because I know I'll find at least two or three good new programs for my Apple. Other than "Imhotep" (a real loser), I've

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.



Outgoing Mail

by Dave Albert

Here we go again, this month's chance to talk back at you, the reader. There are some points to clarify, some tentative good news, and various and sundry miscellaneous ramblings to crank out. Here goes...

The tentative good news is that we may soon be reviving the "Data Base" series started by Mark Pelczarski last September. He is currently working on hammering out some bugs in the Atari version, and says he will send both Apple and Atari versions after all the bugs are cleared up. There's always the possibility that he won't be able to vanquish the bugs, but we are hoping that we may soon resurrect the series. Keep an eye on this column for further developments.


The points to be clarified are about the "magazine of the future" discussed and advertised in the previous issue of **SoftSide**. We intend to go ahead full steam on the project, but we also feel that the ads may have been misleading in one respect. In an undertaking of this magnitude, we feel that we have to tackle the different computer versions one at a time. Since the vast majority of our readers own S-80s, that is the first system that we will develop the enhanced Disk Version for. The Apple and Atari disk enhancements will show up later in the year.

Another point I wanted to mention is that this month we decided to break with a tradition which we have held to since we went glossy, which is that of using photographic covers. This

month's cover is a piece of artwork by a local artist named Bill Giese. We are quite proud of it, and may go to more illustrated covers in the future. Please let us know how you like it.

And while I'm on the subject of covers, we neglected to mention that the previous four cover photographs were taken by Mary Locke, our copy editor/photographer. In future issues we will try to make sure to give credit where it's due for our cover illustrations, be they photographs or otherwise.

I have frequently complained in this column about the total lack of Adventure submissions for machines other than the S-80...well, that is no longer true. We have received ONE such submission, an Adventure written on the Atari. It has yet to be evaluated, but it does contain extensive graphics and the preliminary feeling amongst the editorial sorts is that we may go with it in a future issue. But that doesn't mean that you 6502 types should rest on your rather skimpy laurels...we still would like to see more submissions and fewer complaints about the lack of such programs in the magazine. We can't print what we don't have.

Next month I will try to answer some more questions that have come up in the mail, and perhaps launch another effort to clarify the mysterious goings on around **SoftSide**. Until then, keep those letters and cards coming, preferably along with your latest efforts on cassette or disk. 

ATTENTION AUTHORS

SoftSide magazine, the leader in the field of BASIC software programming for home computer applications, is actively seeking program and article submissions for the more popular home microcomputers, as well as product reviews. This is your chance to make some extra cash and become famous in the process!

We are interested in programs written in BASIC with any alternate language subroutines worked into the program only within the framework of BASIC. Games 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.

We are looking for well-written, informed reviews of all software for the popular home computers for publication in the magazine. Reviews 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 consumer's interest.

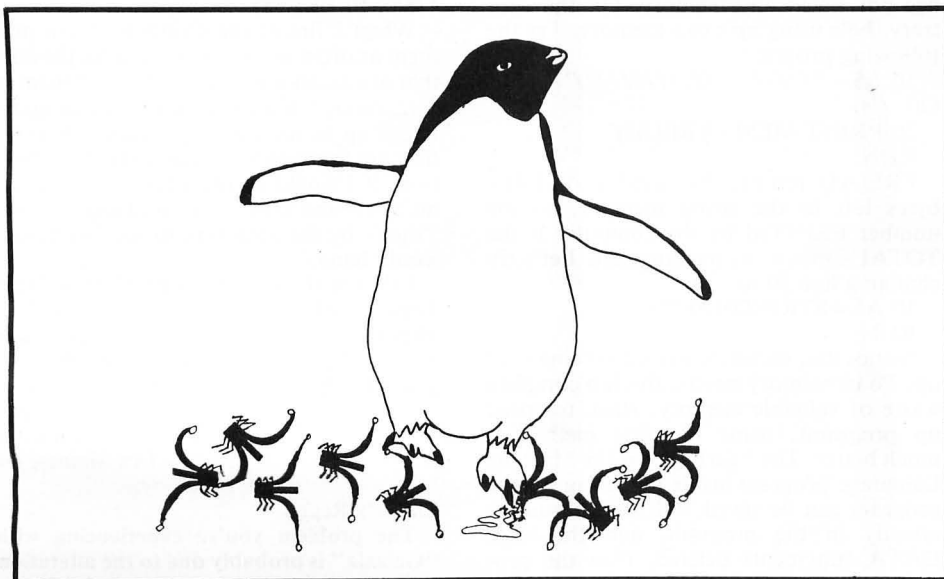
When submitting a program, please be sure to include full documentation of subroutines and a list of variables, as well as a brief article describing the program. All such text, as well as article and product review submissions, should be typewritten and double-spaced. Programs should be submitted on a good cassette or disk, and should function under both Level II and Disk BASIC.

Send to:

SoftSide Publications
SUBMISSIONS DEPARTMENT
6 South Street
Milford, NH, 03055

Be sure to send for our free Author's Guide.

We regret that due to the volume of submissions we receive, we are unable to return your cassettes or disks.



Input

continued from page 6

found every program well-done, and worth the effort to copy. Some of the subroutines have even found their way into other programs that I've put together myself. So far I haven't found any bugs or worms (If I do I may have to take this all back), and everything runs well.

Do I have any complaints, well...just one. Your letter from Brian Yamauchi, Input, May, 1981, is almost word for word a duplicate of what I sent you.

How about doing something so my issue gets here in one piece? Perhaps packing it in a wooden crate, or maybe a stainless steel envelope would help. Have you considered a plain brown wrapper with threats against the postal service if they mutilate it, or perhaps hand delivery? Or maybe just using a better quality paper would help! I like what's inside **SoftSide**, keep up the good work, but let's try to shape up on getting it here in one piece.

Bob Devine
Adona, AR

Dear **SoftSide**,

First, thanks for many interesting hours spent with your articles and programs. Even occasionally having to figure out why the programs won't run as given (remember "BOING"?), and fixing them, has been fun. Mostly. We (the Atari 400 was my son's Christmas present, and sometimes he even gets to use it) especially enjoy modifying programs, to add frills and to learn more about the art of programming.

One problem we've had during modifying/debugging is the speed with which a LIST scrolls up across the screen. Trying to locate a particular statement or to find an error in typing becomes an aggravation, and time is wasted LISTing and RELISTing. Also the language gets bad (you should HEAR that kid). We saw the S-80 program, "WHOA!", in the December, 1980, issue, which didn't help except to provide the impetus to think up a routine to solve the problem for the Atari. To use it, start each program (at least during the debugging stage) with

```
1 GOTO 10
2 FOR I=LL TO UL: LIST I
3 IF PEEK(53775)<>251 THEN 3
4 PRINT "^^": NEXT I: STOP
```

Line 1 lets a RUN command skip over the routine. (If the first regular program line number is something other than 10, GOTO that instead.)

Line 2 scans through all numbers from LL (Lower Line) to UL (Upper Line), LISTing them one by one. (Often it's simpler to replace UL with some number larger than the largest line number in the program.)

Line 3 stops the scanning EXCEPT when a keyboard key is being pressed. (Any key; usually the spacebar is most convenient.)

Line 4 moves the cursor up two lines between each LIST (the up arrows are

ESCAPE CONTROL characters) to avoid spaces between the LISTed lines, and (especially) to prevent spaces being added for "empty" numbers. Line 4 then goes to the next number, and STOPs when finished.

To list the entire program (with UL replaced by a suitably large number), type LL=1: G.2 and hit RETURN. Brief touches on a key will add one program line at a time, while continuous pressure will scroll lines up (but conveniently slower than the normal LIST). Release the key and the displayed lines will wait until you're ready to continue.

To edit the program you have to BREAK, but you can continue from any desired point by redefining LL. Note that LL doesn't have to be a program line number. Also, if there are big gaps in program line numbers, for example between subroutines, it's faster to BREAK and increase LL than to wait while the routine scans through several hundred "empty" numbers.

Of course this routine could be added at the end of a program, or in the middle — but G.2 is easier to remember, and to type, than something like G.5460.

Maybe all the old-timers already have something like this figured out, but, if so, they aren't telling us beginners. Perhaps it can help reduce eye strain and frayed nerves.

Let me end with a request: the One Liners are great for learning new ways to exploit the Atari's graphics capabilities. . . but can you print an explanation of how the one by Kinzebach (May, 1981, p.35) does what it does? It's frustrating to enjoy the result, but not to know what's going on: Keep up the good work on **SoftSide**.

Dixon Stroup
Kailua, HI

Dear **SoftSide**,

I just received my April issue of **SoftSide** and noticed that there was a slight error in the "Programming Hints" article by Shane Causer. Although Mr. Causer has cut down the screen "garbage" by redefining A\$, he has not saved any memory. On the contrary, he's using up extra memory. Try the following program:

```
10 A$ = "////////////////////"
(20 '/'s)
20 PRINT MEM + FRE(A$)
RUN
```

FRE(A\$) returns the number of FREE bytes left in the string memory, so the number PRINTed by the computer is the TOTAL amount of memory left. Let's try changing line 10 to:

```
10 A$ = STRING$(20,75)
RUN
```

Notice that six whole bytes are being used up. To us memory misers, this is a complete waste of valuable memory. Also, to speed up programs, using the first method is much better. The "garbage" is the Machine Language program inside the string, so the program can be saved with the subroutine already in the program, and the large DATA statements deleted. (See the pro-

gram "Cards" in the the November, 1980, **SoftSide**, pg. 57)

Also, I have found a slight problem when typing in these strings (as in the first line 10, above.) Sometimes I might slip and type the wrong number of '/'s. So, instead of taking a risk, I now type:

```
10 A$ = "12345678901234567890"
```

This is simple to type, and it is easy to tell the length. I find this necessary because I once was thrown to "MEM SIZE?" because of a typing error.

For those twisted people out there who like doing strange things to their computers, try the following: (Do not do this if you have not saved what is on the computer at the moment)

```
POKE16405,0
```

Once you find that this disables your keyboard, the only thing you can do is turn the computer off. (I warned you not to have a program on your computer. Don't blame me if you just lost three hours of typing.)

One last twisted, demented thing to do to your computer:

```
10 POKE16445,8:PRINT"ABCDEFG..."
20 FORX=0TO1000:NEXT
RUN.
```

You figure it out.

Thomas Andrews
Chicago, IL

Dear **SoftSide**,

Please make arrangements with SOMEONE to take your Apple programs and bench check them for errors. I said "someone" but please use at least three people. Give them copies of the final printout to be printed in **SoftSide**. If there are errors, they will find them.

I have just finished two weeks trying to debug "Galaxia". The first problem I found was line 90. It may be a quirk of DOS 3.1, but I had to reverse LOMEM and HIMEM. I also found it necessary to find the correct HI/LOMEM with the following coding:

```
HI — PRINT PEEK(116)
*256 + PEEK(115)
LO — PRINT PEEK(106)
*256 + PEEK(105)
```

When I fire at the Crinoids, I can miss them as often as I wish or I can hit the ones that are attacking. If I hit ANY of them in formation, I am in trouble. The program hangs up as soon as one reaches the point they are to reposition back up to the formation. If I would let the attackers crash into me to the last base, it would hang. If I was "shot" by the attackers to the last base it would hang.

I do not think I have a problem with my keying. All of the program has been checked at least three times (lines 1130—1330). The remainder of the program has been checked at least six times.

Ken Asmussen
Des Moines, IA

Editor's Reply:

The problem you're experiencing with "Galaxia" is probably due to the alteration

you mentioned to line 90. The purpose of setting LOMEM and HIMEM is to protect certain areas of memory; if these values are changed, then the memory areas in question may be over-written with unwanted information.

Applesoft stores the actual program lines in memory starting at address 2052, and then automatically sets LOMEM just above the program. LOMEM, then, is the lowest address available for variable storage. HIMEM is automatically set at the top of RAM memory for a system without disk; or, with a disk, it's set just below the DOS which is loaded into the top 10.5K of memory. HIMEM is the highest address available for variables. If you PEEK into memory locations 105-106 and 115-116, as you suggested, the values you find will be those set automatically by Applesoft.

In "Galaxia", however, those values need to be reset to protect two areas of memory. One area is the second Hi-Res graphics page, which occupies memory locations 16384-24575. Setting LOMEM just above this keeps the program from using it as variable space. The second area is that which contains the various Machine Language routines POKEd in by lines 1110-1330. These routines occupy 616 bytes of memory starting at 31744, so HIMEM must be set below that. If it isn't, the routines may be overwritten by variables and the system will hang up or crash when the routines are called.

I can't see why DOS 3.1 should have any problem with the specified LOMEM: and HIMEM: statements, since they are Applesoft commands. You would need to have a 48K machine, however, to use "Galaxia" with the DOS booted. I assume you do, since you indicate no problem with using HGR2, which is not available to a 32K disk user either.

If you do find an error in the published listing, please let us know. As with all our listings, this one was dumped onto a lineprinter directly from a working version of the program; if a character was misprinted or a line dropped, no one has found the error yet.

Dear SoftSide,

I loved the "Lunar Mission" by Matt Rutter, in the May, 1981, edition of SoftSide, however I added some lines for amateurs that will abort your landing procedures, but will leave the same amount of fuel that you had before you aborted. To abort the mission, push the fire button on the joystick, while pushing up on the joystick.

The additional lines read as follows, but the lettering in the print statement in line 1100 should be typed in reverse graphics.

```

234 IF STRIG(0)=0 THEN 1000
1000 FOR Q=1 TO 4:POKE
656,1:POKE 657,2:"MISSION
ABORTED": SOUND 0,0,0,0:FOR G=1
TO 70:NEXT G
1100 POKE 656,1:POKE 657,2:
"MISSION ABORTED"

```

```

1200 FOR K=1 TO 30:SOUND
0,10,10,10:SOUND 1,20,10,10:SOUND
2,30,10,10:SOUND 3,40,10,10
1300 FOR K1=0 TO 3:SOUND
K1,0,0,0:NEXT K1:NEXT K:NEXT
Q:GOTO 10

```

Evan Price
Roslyn Heights, NY

Dear SoftSide,

As sponsor of a high school computer club, I can speak for all our members and say that we think very highly of your magazines. Primarily organized for entertainment, the members especially enjoy the games you publish.

The "Collision" game by Mark Pelczarski in November, 1980, is especially outstanding and gets much playing time. I have altered it slightly so that it models the arcade version more closely. The following listing keeps a short sequential file called HIGH that holds the scores and names of the top 10 scorers.

I also included a listing of our top 10 and would like to hear from others who have exceptional scores. Most of these scores have been into the fourth rack and two have been in the fifth rack.

Keep up the good work, it sure helps teachers like myself to keep the interest going.

Before running "Collision" with my changes, run this file initialization program:

```

10 D$ = CHR$(4)
20 FOR J = 1 TO 10:HS(J) = 0:HS$(
(J) = "": NEXT J
30 PRINT D$;"OPEN HIGH"
40 PRINT D$;"WRITE HIGH"
50 FOR I = 1 TO 10: PRINT HS(I):
PRINT HS$(I): NEXT I
60 PRINT D$;"CLOSE HIGH"

```

Lynn N. Leopard
Chillicothe, MO

```

1 HOME
5 D$ = CHR$(4)
7 DIM HS$(12),HS(12)
10 REM COLLISION
20 REM C. 1980, MARK PELCZARSKI

25 PRINT D$;"OPEN HIGH"
30 PRINT D$;"READ HIGH"
40 FOR I = 1 TO 10: INPUT HS$(I)
: INPUT HS(I): NEXT I
45 PRINT D$;"CLOSE HIGH"
46 GOSUB 4000
200 IF XY = XC AND YY = YC THEN
900
205 IF RK > 2 THEN IF XY = XD AND
YY = YD THEN 900

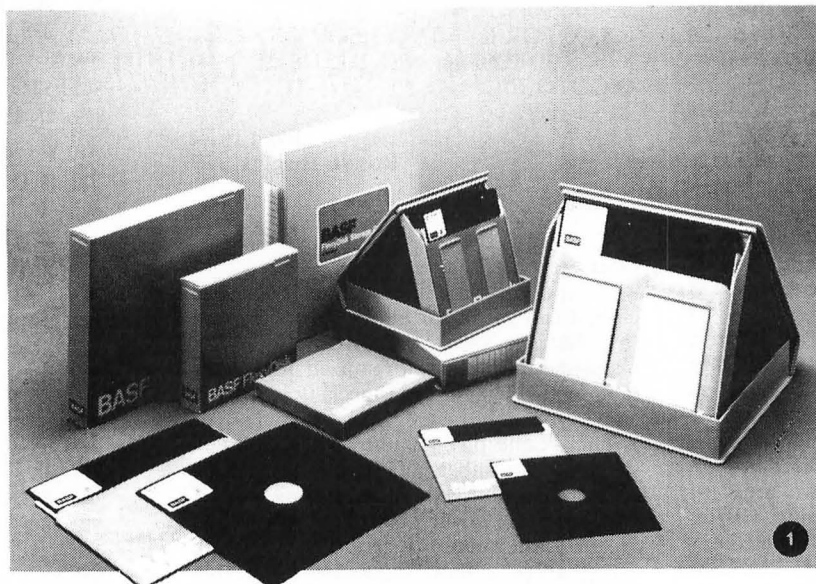
```

```

900 REM CHECK NEW SCORE FOR TOP 10
901 TEXT : HOME : PRINT "YOUR SC
ORE WAS ";SC
904 QQ = 0
905 FOR II = 1 TO 10
906 IF SC < HS(II) THEN 929
910 X = HS(II):X$ = HS$(II)
912 HS(II) = SC
913 IF II = 1 THEN PRINT "CONGR
ATULATIONS! NEW HIGH SCORE!
!!": INPUT "PLAYER'S NAME: "
:HS$(II): GOTO 915
914 PRINT "NEW TOP 10 SCORE!!": INPUT
HS$(II)
915 IF HS$(II) = "" THEN 913
916 IF II < 9 THEN 923
917 IF II = 10 THEN 927
918 IF II = 9 THEN 926
919 NEXT JJ
920 HS(II + 1) = X:HS$(II + 1) =
X$
922 II = 10
923 FOR JJ = 10 TO II + 2 STEP -
1
924 HS(JJ) = HS(JJ - 1):HS$(JJ) =
HS$(JJ - 1)
925 NEXT JJ
926 HS(II + 1) = X:HS$(II + 1) =
X$
927 II = 10
928 QQ = 1
929 NEXT II
930 IF QQ = 0 THEN 955
935 PRINT D$;"OPEN HIGH"
940 PRINT D$;"WRITE HIGH"
945 FOR II = 1 TO 10: PRINT HS$(
II): PRINT HS$(II): NEXT II
950 PRINT D$;"CLOSE HIGH"
952 GOSUB 4000: HOME
955 PRINT "PLAY AGAIN (Y/N) ?":
GET A$
965 IF A$ = "Y" THEN 1013
970 HOME : END
4000 REM SUBROUTINE TO PRINT HO
NOR ROLL
4001 HOME
4005 PRINT SPC(10);"C O L L I
S I O N"
4010 PRINT SPC(10);"TOP 10 HON
OR ROLL"
4015 PRINT "RANK","NAME","SCORE"

4020 FOR I = 1 TO 37: PRINT "-";
: NEXT I
4022 PRINT
4025 FOR I = 1 TO 10
4026 X = 1
4027 IF I = 10 THEN X = 0
4030 PRINT I; SPC(10 + X);HS$(I
); SPC(21 - LEN(HS$(I)));
HS(I)
4040 IF I = 10 THEN 4060
4050 PRINT
4060 NEXT I
4065 GET A$
4067 HOME
4070 RETURN

```



We have all the supplies
you need for your micro.

call us today!

BASF Diskettes

- Box of 10, 5 1/4" Single Density
- #15-DKBSD \$34.95
- Box of 10, 5 1/4" Double Density
- #15-DKBDD \$42.95
- Box of 10, 8" Double Density
- #15-DKB8 \$49.95

3-M Scotch Diskettes

Encased in a rough jacket which resists handling damages. 100% certified error-free performance. Low modulation provides better signal stability.

- Box of 10#15-DKSC \$39.95

Digital Cassettes

Premium quality digital recording tapes. These five-screw cassettes come packaged in boxes of ten. They are offered in 10 and 20 minute lengths.

- #15-C10 \$6.95
- #15-C20 \$7.95

Diskette Head Cleaning Kit

3-M Scotch 7400 head cleaning kit is simple and easy to use. You simply saturate the write head cleaning fabric in the cleaning diskette with the cleaning solution, insert the diskette into the drive and turn it on. The rotating cleaning fabric alternately wipes the heads with the solution and the dry surface, removing contamination from the read and write head. Each kit contains two cleaning diskettes which will allow you a total of 100 cleanings.

- 15-SHCK \$29.95

Soundware

Soundware adds a whole new dimension to your computer games. Programs come alive with laser sounds, bounces, clicks, sirens, bird calls, music notes, tunes, and whatever else your imagination dreams up. Just slip in two AA batteries, plug into your computer, and plug into your computer, and have fun. Soundware Software programs are also available to enhance the enjoyment of your computer.

- TRS-80 Mod I/III #26-SNDWR .. \$29.95

Diskette Library Case

Allows you to store up to 15 mini-diskettes in a strong molded plastic carrier.

- 15-DKBX \$5.00

Floppy Disk Saver

PREVENTS: Computer drive's cleaning clamping hub from tearing disk's center hole; coating removal, scuffing, dimpling; and data loss caused by improper rotation.

- #15-FSAV \$14.95
- #15-FSAVR Refills \$7.95

Disk Storage Box

Static-free design protects up to 50 mini-diskettes. Adaptable to hold cassette tapes. Boxes are stackable. Smoke-tinted cover, three sectional dividers, and durable molded plastic.

- #19-06 (2 lbs.) \$19.95



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



Calendar of Events

August 24-27

Software Design, Reliability, and Testing Seminar
Sheraton Motor Inn, Lexington, MA

A four-day seminar for engineers, programmers, and technical managers. Concepts and techniques for developing and testing reliable, cost-effective software and management concerns are discussed. Tuition is \$600 which includes course notes, refreshments, lunch, and an evening reception.

Contact: Institute for Advanced Professional Studies,
One Gateway Ctr., Newton, MA 02158, (617)964-1412.

August 24-28

Seventh International Joint Conference
On Artificial Intelligence
University of British Columbia, Vancouver,
British Columbia, Canada

Conference will examine computer applications of medical diagnosis, computer-aided design, robotics, programmable automation, speech understanding, vision, etc.

Contact: Louis G. Robinson, American Association for
Artificial Intelligence, Stanford University,
P.O. Box 3036, Stanford, CA 94305, (415)495-8825.

August 25-28

Vector and Parellel Processors in Computational
Science Conference
Chester, England

Conference will concentrate on hardware, software, algorithms, applications and case studies concerning vector and parallel processors.

Contact: Mrs. S.A. Lowndes, Science Research Council,
Daresbury, Warrington, WA4 4AD, England.

August 26-29

Fifth Annual National Small Computer Show
New York Coliseum, New York, NY

Daily lectures and a five-hour seminar will also be presented daily for executives who would like an overall introduction to understanding, buying, and using computers in business. Registration fee for the show is \$10 a day. The seminar fee is \$200 per person, including all materials and show registration.

Contact: National Small Computer Show,
110 Charlotte Place, Englewood Cliffs, NJ 07632
(201)569-8542.

August 26

Boston Computer Society Meeting
New England Life Hall, Boston, MA

Meeting begins at 7:30. Various topics of general interest to microcomputer owners and users are discussed. Meetings are every month.

Contact: Julie Kuhn, Three Center Plaza, Boston,
MA 02108 (617)367-8080.

August 28-30

Personal Computer Arts Festival
Philadelphia Civic Center, Philadelphia, PA

Held in conjunction with Personal Computing '81 Show. Demonstrations and talks on microcomputer music synthesis, computer-generated visual art, and other computer-based creations.

Contact: PCAF-81, Box 1954, Philadelphia, PA 19105.



Dragonquest!

In a desperate race against the sun you search for SMAEGOR Monarch of Dragonfolk, who has kidnapped the Princess of the Realm and holds her in a distant and unknown place. In a quest for Honor and glory, you must search the land, seeking out the tools needed for the ultimate confrontation. On The River Delta, in the abandoned Temple of Baathteski, Goddess of the Blade, everywhere, clues abound. But WHERE is the Princess?

Now, as never before, the genius of CHARLES FORSYTHE shines in this new machine language ADVENTURE. DRAGONQUEST! Can YOU save M'lady from the iron clutches of SMAEGOR?

S-80 Level II 16K Cassette #26-221001T . . . \$15.95
S-80 32K Disk #26-221001D \$21.95

From
The Programmer's Guild


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

THESE HARMLESS DISCS CONTAIN ENOUGH EXPLOSIVES TO SINK A DESTROYER. OR WIPE OUT AN ENTIRE BATTALION.

These mini-floppy discs for your Apple® computer represent the culmination of our extensive R&D efforts to bring you state-of-the-art strategy games in submarine and land warfare: **TORPEDO FIRE** and **OPERATION APOCALYPSE**.

These power-packed games from Strategic Simulations Inc. are designed with the same loving care and uncompromising standards that have made **COMPUTER BISMARCK** — our flagship game — a phenomenal success hailed by critics and enthusiasts.

TORPEDO FIRE™ takes you to the high seas and murky depths where you'll play both hunter and hunted in the desperate battle between submarines and convoy escorts.

You are given the sophistication of simultaneous order execution and realistic sighting rules. You can challenge another player or engage in solitary warfare where the computer plays the submarines.

Create your fleet from 30 ships of the four major navies (all rated with historical accuracy for speed, weaponry, and maneuverability) — or design the ships to your own specifications. Make up any multitude of scenarios — day or night actions, single- or multiple-ship battles.

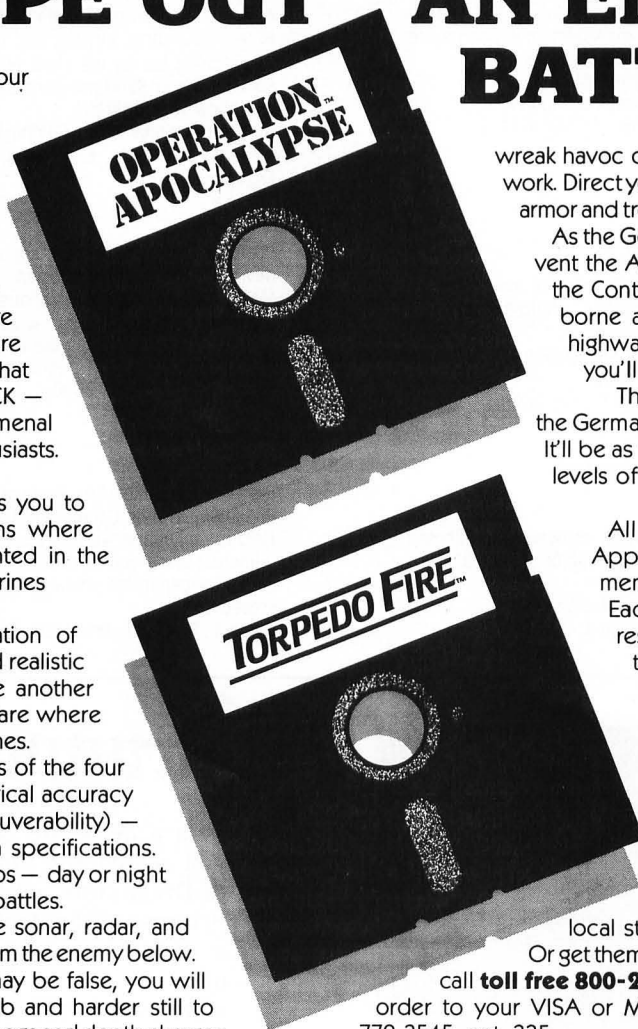
As the escort commander, use sonar, radar, and your eyes to protect the convoy from the enemy below. Since the sightings you receive may be false, you will be hard pressed to track the sub and harder still to force it to surface with your hedgehogs and depth charges.

As the submarine commander, you can make full use of the remarkable computer-generated Hi-Res periscope view to locate your prey. You must then destroy the convoy, attack or evade the escorts — all with utmost stealth, lest the seas become your watery grave.

OPERATION APOCALYPSE™ carries you to the Western Front, circa 1944. You have the opportunity to re-enact the various facets of the Invasion of Europe in four separate scenarios, each offering different victory conditions, personnel, ordnance, and terrain.

OPERATION APOCALYPSE uses a revolutionary terrain and movement system designed to give you easy and complete control over your forces: engineer, infantry, artillery, and armor units. The action takes place on a 7-by-18 hexagon mapboard dotted with hills, rivers, bridges, forests, and towns. For further battlefield realism, the game also offers hidden movement.

As the Allied General, you can order off-screen artillery bombardment to soften up German resistance. Or call upon airborne landings behind enemy lines to capture key bridges or to



wreak havoc on the enemy's communications network. Direct your engineers to build bridges so your armor and troops can roll towards their objectives.

As the German High Command, you must prevent the Allies from gaining a firm foothold on the Continent by quickly wiping out their airborne and amphibious landings. Sever the highways and bridges to Germany, and you'll cripple the Allied advance.

The computer is ready to take you on as the Germans anytime you want a solitary game. It'll be as tough as you like since you have four levels of difficulty to choose from.

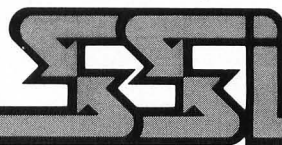
All you need to play both games are an Apple II with Applesoft ROM card, 48K memory, and a mini floppy disc drive. Each for \$59.95, both come with their respective program disc, a rule book, two mapboard cards (for plotting secret strategies between moves), and various player-aid charts.

Without a doubt, **TORPEDO FIRE** and **OPERATION APOCALYPSE** represent the finest computer wargames available, head and shoulders above their competition.

So why wait? Hurry down to your local store and get your copies today!

Or get them directly from SSI. Credit card holders, call **toll free 800-227-1617, ext. 335** and charge your order to your VISA or MASTERCARD. In California, call 800-772-3545, ext. 335.

To order by mail, send your check to: Strategic Simulations Inc., Dept. S2, 465 Fairchild Drive, Suite 108, Mountain View, CA 94043. All our games carry a 14-day money back guarantee.



SSI's other games for your Apple:

COMPUTER BISMARCK, * \$59.95	COMPUTER CONFLICT, \$39.95
COMPUTER AMBUSH, \$59.95	COMPUTER AIR COMBAT, \$59.95
COMPUTER NAPOLEONICS, \$59.95	THE WARP FACTOR, \$39.95
COMPUTER QUARTERBACK, \$39.95	CARTELS & CUTTHROATS, \$39.95

* Also available for the TRS-80 — Disc, \$59.95; Cassette, \$49.95

As part of our demanding standards of excellence, we use **maxell** floppy discs.

TRS-80 is a registered trademark of Tandy Corporation.

Apple is a registered trademark of Apple Computer Inc.

About This Issue

It's summertime, the living's easy, the weather's hot, the mint juleps abound, and we munchkin types feel that underground may be the best place to go to beat the heat. So...

☐ We proudly present "Quest", an effort by Brian Reynolds, dexterously (deftly, at least) translated by Mssrs. Bouchard and Zett. This program permits the (fool)hardy adventurer to wander amongst the labyrinthine passages of an underground world, where the sun don't shine, as they say. "Apschai" veterans may recognize the style, but there are some added twists. See for yourselves.

☐ And just when you thought it was safe to go back into the ROM, we present "Shark", a lovely little Apple maritime game where you get to be the voracious predator and your victims are cute little fish. We munchkins have named all of the fish after our bosses, but they won't know about that unless they read this, which we doubt they will. The program was written by Mark Cross, and we hope to see more of his stuff in future issues.

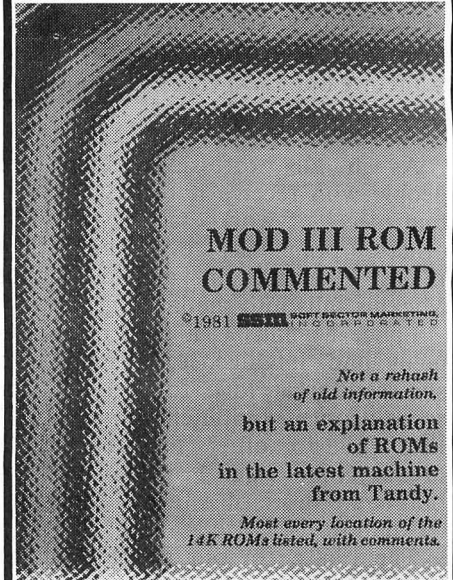
☐ Another program presented for all of the systems we support is "Battlefield", originally written by Joe Humphrey, translated for the S-80 and the Atari by none other than Jon Voskuil, that fellow of questionable Dutch descent who brought you the delights of mathletics. The man's a regular polyglot.

☐ Now that you S-80 types understand what a VARPTR is, thanks to J.T. Phillipp, M.D., you can read his next installment and figure out how to use one. If you didn't read the first installment, tough. We can at least assure you that a VARPTR has nothing to do with the interstellar drives on the U.S.S. Exitprise, Cap'n Kirk notwithstanding.

☐ And Ol' Granite Knoggin, our resident hardware fount of wisdom has a treat for you Model I owners...a cheap and simple video reverse modification that will ease eyestrain, enhance the graphics in virtually any game we've played (except spin the bottle), and generally make your computer a source of wonderment for all your compatriot S-80 owners.

☐ Plus, Lance Micklus continues in his quest for aiding folks in the software business; Joan Truckenbrod tackles reflective symmetry in Apple graphics; there's another installment in that series of seemingly endless columns, or is that a seemingly endless series of columns?, written by that sensuous fellow "J". And there's more, but you'll just have to read on to find out about it! So until next month, a merry munchkin farewell. And don't worry about that truck incident mentioned last month, it was just a ripoff from an essay written by Saturday Night Live's own Mr. Mike. Cheerio!

Do you reap only mysteries from your TRS-80 ROMs?



Your book has arrived

Mod III Rom Commented is the book Mod III Users have been waiting for — it gives the location of most of the 14K ROMs in the Mod III with commentary on them.

#65-275001B \$22.50



TSE-HARDSIDE

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

"We buy and sell used computers & peripherals."

Save!

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

ATARI 400, 8K RAM.....	\$275.00
CENTRONICS 730/LP-II.....	\$325.00
CENTRONICS 799/FRICTION.....	\$325.00
CENTRONICS 799/TRACTOR.....	\$350.00
HAZILTINE 1410 TERMINAL.....	\$350.00
RS 32K EXPANSION INTERFACE.....	\$310.00
DATA SEPARATOR.....	\$15.00
DOUBLER.....	\$70.00
MOD-I SYSTEM DESK.....	\$50.00 +
TRS-80 MOD-I, 16K, L-II.....	\$625.00
TRS-80 40-TRACK DISK DRIVE.....	\$250.00
DUAL 40-TRACK DISK DRIVE.....	\$475.00
CENTRONICS 779 (DEMO).....	\$400.00

QUANTITIES ARE LIMITED and SOLD ON A FIRST-COME-FIRST-SERVE BASIS. All used equipment is sold with a 30-day HARDSIDE warranty. All equipment is in working condition. Some cases may have some scratches from normal useage. + Prices do not include shipping charges.



by I. Alexis Adams

Lance Micklus — Magic Man

This month we have a guest column in place of Scott's usual scribblings. Scott will return next month; meanwhile, we hope you enjoy this one from the other half of AI.

As in all professions, there are people who receive notoriety for certain pieces of work that they do. For others, practically everything they release becomes a classic. Lance Micklus falls into the latter group.

Lance Micklus is a man of excitement and intelligence, a man who shows a great love for people. For some people, computers are a way of life, while for others they represent a dream come true. For Lance, computers are more of a fate...as he says, "I fell into it, twice."

The 1960s, for our people and for our country, constituted a time of many new beginnings. Rioting, fighting, and rebellion filled the streets. In Lance's case, the '60s marked the beginning of his career, although not even he knew it then.

It began in the summer after he had finished high school, before he was to start college in the fall. IBM was offering summer jobs to the children of employees and Lance decided to take a position. For the next three months he was a computer operator. The following year all they had available was a full-time position, so for the following year and a half he worked for IBM in that capacity. It was at IBM that he learned Machine Language.

Lance was still attending college when he worked for IBM. In 1969 he graduated, and, rather than returning to a career in computers, he decided to follow another love: radio. He joined up with a radio station and worked there for a while, and then proceeded to go to work for a television station. The TV station used computers, and the knowledge of Fortran that Lance had picked up in college was put to good use. Lance fell ill for about six weeks while he was working for the TV station and used a terminal at his home for work during that time. The terminal at home helped him to make the decision to go back to using a computer.

Lance had heard about the Radio Shack computer and decided to look

into it further. He soon purchased one with the hopes of using it as a terminal. When he bought it, he was told that a super terminal package was soon to be released. He never imagined that he would come to write the terminal packages that would become the industry standards.

Lance's path to fame began when he answered an ad in *Kilobaud* for TSE's lending library, a legitimate lending library where one would send in \$10 and get a tape with a program on it. The tape would be leased out, and part of the money would be used to pay royalties to the author. That idea never fully worked out, but it still gave Lance his introduction to Roger Robitaille the founder and publisher of *SoftSide*. The association was to serve both of them quite well in the future.

"What the Market Needs is Something New, Original and Creative."

"It's hard to make a living just writing software," Lance says. He points out that there aren't many people, if any, that are successfully doing it. Some authors have also branched out into producing and marketing their products. (This is a topic which Lance has covered in the past few installments of his bimonthly column in *SoftSide*...Ed.)


When asked what words of wisdom he might have for up-and-coming authors, Lance notes that "The problem I see with programs now is that they are imitations of others seen elsewhere." For instance, Adam Osborne wrote an accounting package many years back. Companies are now buying the rights, doing small modifications and conversions, and selling it. One of Lance's current projects is writing his own accounting system, using his own ideas. What the market needs is something new, original and creative.

Lance's recent marriage to his lovely bride, Dianne, hasn't hindered his career a bit. She is very supportive of him, yet is not directly connected with

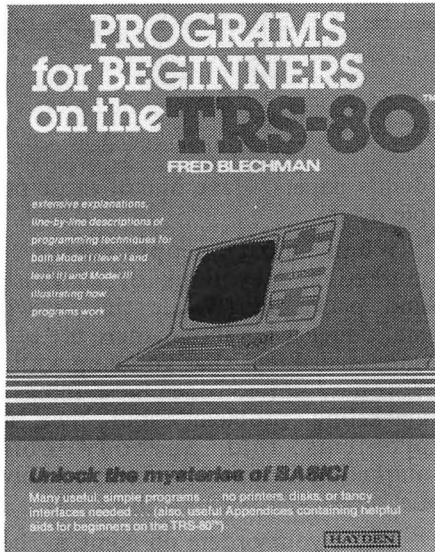
his business. I would like to take this space to say that I think Lance is very lucky to have such a supportive partner. While Lance and Dianne honeymooned in Atlantic City, he received the inspiration for his latest program. Not being a gambler, and seeing all the gambling going on, he became very interested in the game of Craps. They learned all that they could about the game and took a gambling table home. Lance scanned the industry (computing) for Craps games and was left dissatisfied with what he found. As he did with his smart terminal programs, he decided to write his own version of Craps for the color computer. He hopes to have it available for the S-80 Model I and Model III soon. (For more on the development of this infamous program, see Lance's past columns...Ed.)

When I asked him where his inspiration comes from, he said his hobby of magic helps him a lot. His residence in past years was in Michigan, some 15 miles from Colan, which is the magic capital of the world. Lance uses his magic to entertain people. He feels that the programs he writes on the computer are similar in that their purpose is to entertain. He never feels like he is working on a machine. He feels that his computer is the instrument which allows him to implement his creativity to create his software masterpieces.

Lance's other hobby, photography, has equally helped him in his career. The photography methods used for graphics filming at the TV station are much like what is used on the color computer. While what he photographs in life is different than his game graphics, his experience certainly helps him in his computer endeavors. Lance's love for people shines through in his photographs. While his shots of scenery are photographically correct, they don't shine with creativity like his shots of weddings and nudes which are his specialty.

What lies ahead not even Lance and Dianne know. I am certain, however, that anything that Lance does will be given his best effort. If his work was a star, it would be shining all over the world, as bright as gold. 

New Books from TSE HARDSIDE



Programs For Beginners on the TRS-80

by FRED BLECHMAN

Here's a valuable book of practical and interesting programs for home use that can be understood and used immediately by the beginner in personal computer programming. You'll learn step-by-step how 21 sample TRS-80 programs work. Program techniques are described line-by-line within the programs, and a unique Matri-Dex matrix index will enable you to locate other programs using the same BASIC commands and statements. Each program includes a detailed description, a complete listing, an explanation of what the program does, and instructions for modification.

HAYDEN BOOK COMPANY, Inc.

Order #65-21004B.....\$8.95

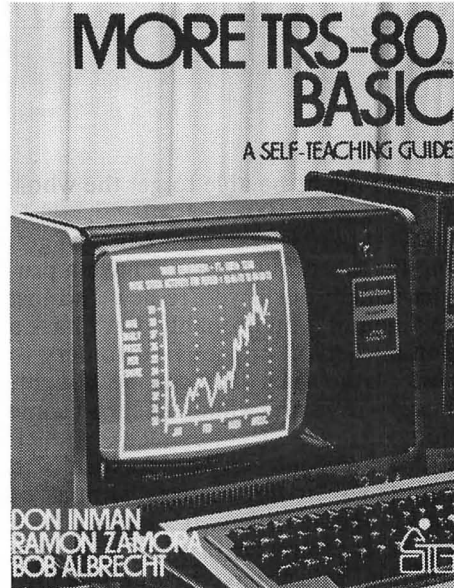
Apple Machine Language

by DON INMAN & KURT INMAN

Learn machine language programming in practically no time at all! It's easy with this ALL NEW guide to the APPLE computer. If you already know BASIC, you're on your way to developing real expertise on the subject. This book combines color, graphics, and sound with clear-cut demonstrations to help you learn — FAST! You'll start off simply, using BASIC statements like POKE, PEEK, and CALL to enter and execute machine language programs. Then you'll develop a BASIC Operating System, entering and executing your machine language programs from there. The next step will have you using an Apple System Monitor to program directly and save the time usually needed to interpret BASIC statements. Finally, you'll learn how to use the Apple Mini-Assembler to avoid the troublesome details involved with direct machine language programming.

RESTON PUBLISHING, Inc.

Order #65-211001B.....\$12.95



More TRS-80 BASIC

by DON INMAN/RAMON ZAMORA/
BOB ALBRECHT

Here is an exciting new interactive programming guide to advanced functions. A sequel to the authors' highly popular book **TRS-80 BASIC**, **More TRS-80 BASIC** helps you build on your fundamental knowledge of Level-II BASIC with scores of practical applications that are both recreational and educational. It explains all TRS-80 Level-II BASIC statements, commands and features not covered in their previous best seller. Along with an exploration of the PEEK and POKE functions of the TRS-80 Model-I, you'll get a clear demonstration of several different methods to display graphics, including little known "super-graphics" techniques. You'll explore vital concepts in information processing and master file handling techniques that apply to both cassette and disk files. You'll also discover many "universal" applications easily adaptable to specific problems.

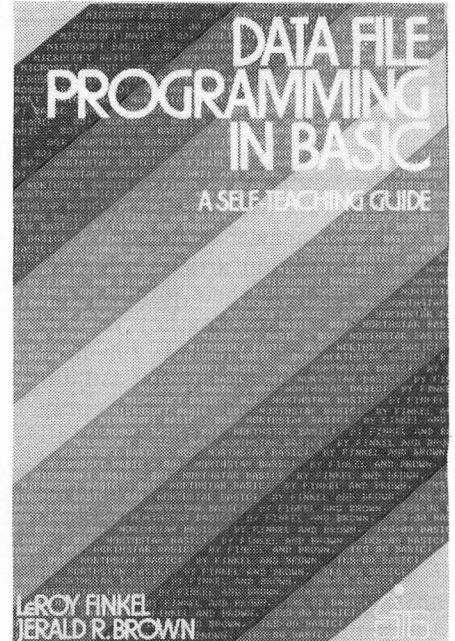
JOHN WILEY & SONS, Inc.
Order #65-233002B.....\$9.95

Apple-II User's Guide

by LON POOLE

The **Apple-II User's Guide** is the key to unlocking the full power of your Apple-II or Apple-II Plus computer. The extensive user's guide will help you program in two versions of BASIC using sound, color, and graphics to full effect. This book contains detailed information on the disk drive and the printer. Includes tips on advanced programming topics. Fully describes how to use the Machine Language Monitor. Shows how to use high resolution graphics with Integer BASIC. Provides a compendium which thoroughly describes every BASIC statement, command, and function. It's all here, in the **Apple-II User's Guide**, thoughtfully organized and easy to use.

OSBORNE/McGRAW-HILL.
Order #65-255001B.....\$15.00



Data File Programming In BASIC

by LeROY FINKEL & JERALD BROWN

Data files are the key to effectively using microcomputers for virtually all business, classroom management, professional, and sophisticated home applications. Yet surprisingly few micro users understand how data files work, and fewer still are able to set up and maintain them for their own use. This easy-to-follow book is the first self-instructional manual designed for the thousands of business and professional users and computer hobbyists who want to add this powerful tool to their computing capabilities. It shows you — in comprehensive and comprehensible detail — how to program and maintain data files on microcomputers, and how to write data file programs that are readable, efficient and useful. This clear nontechnical book leads you at a comfortable pace through each step involved in data file programming and maintaining data files, with dozens of sample programs and practical advice to smooth your way.

JOHN WILEY & SONS, Inc.

Order #65-233001B.....\$9.95




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



My Side of the Page

by Lance Micklus

Getting a Bit Serious—Part 4

I suppose you're all wondering how "The Mean Craps Machine" is making out. Well, the answer is that it isn't. Keeping in mind that articles are written in advance of their publication, Part 3 of this series has not yet appeared in **SoftSide** and I am now writing Part 4. This means that as I am writing this right now, the only persons who know anything at all about "The Mean Craps Machine" are Dennis Brent, Scott Adams, whoever reads article submissions at **SoftSide**, my wife, my son, and the mouse. In short, nobody really knows about my "Craps" game except for a few friends and the family. I'm sure many of you readers can identify with my situation and have been in the same boat. What we need is some marketing and that's what I'm going to cover this month.

Basically, there are two ways to market a program. The first method is to do it yourself and the second is to have somebody else do it for you. Let's consider each of these options.

DOING IT YOURSELF

The first step in marketing something is to let people know exactly what it is you have to sell. There are various ways of doing this — some are cheap and some are expensive.

If you have some type of communications capability, you might advertise your program on any of the various bulletin boards. Your only expense is that of a phone call plus the connection time, if any.

Another method is to submit press releases to the various computer magazines. Many magazines have a section on new software. You might be able to get a free plug there. If you're really lucky, somebody might even write an article about your program to give you more free exposure.

If you're willing to spend some big dollars, you can buy some advertising space at roughly \$1,000 a page. Of course, you don't have to buy a full page ad but I mention the figure to give you some feel for the costs you might incur; but you're also going to have to add to that price the cost of getting camera-ready copy made. Most cities have ad agencies which can do that kind of work for you. If it's a simple

ad, you might be able to get the whole job done for \$100.

Direct mail is another method of getting the word out that you have a product to sell; but if you're new and just starting out, you probably don't have a mailing list of previous customers to work with. Such lists can be bought, though, if you're willing to spend some money.

One excellent way to advertise is to include advertising material with all of your outgoing orders. It's almost like a free ride. However, if this is your first program, you'll have to rule this idea out.

If you sell to retail stores, you can get a lot of free advertising. By becoming part of their line, you automatically get put in their catalog and might even get a spot in the magazine ads.

“One excellent way to advertise is to include advertising material with all of your outgoing orders. It's almost like a free ride.”

Besides advertising, the other obvious problem is packaging. You're going to have to make tapes or disks and include some kind of instructions for their use. Tapes and disks are easy to duplicate because all they require is a computer — something you already have. Instructions are more of a problem. Most people don't have their own photocopiers. Also, many photocopiers don't make nice clean images that look like offset printing. One way around this is to go to one of those speedy print shops. However, it's going to cost you around \$100 to \$200 dollars to get a stack of manuals printed. That assumes camera-ready copy. If you don't have a good word processor and letter quality printer — or no printer — you can usually borrow a decent typewriter from someone and type the thing. Better yet, if you live in a college town, there are always people who sell typing services for stu-

dent's papers. Have one of those people do it.

When most people think of marketing, they think of advertising and packaging. While this is what marketing is all about, there are other unrelated things that go along with it. You may not consider these things as marketing problems, but they are problems that anybody doing marketing is going to have to solve. Let's consider some of these.

When your orders do come in, how are you going to get the money? If the customer sent you a personal check, it might bounce. If you wait for the check to clear the bank, then your order system gets complicated. A lot of people will want to pay with plastic cards. It's easy to become a VISA/MasterCard merchant if you're a business. But, as an individual, the bank may give you a tough time.

The telephone is another source of problems. What happens if somebody calls with a question and your six-year-old kid answers the phone? What happens if they call while you're at work and they leave their work number to call back at? Are you going to sound cheap and call them back collect or are you going to pay for the call yourself?

Speaking of the telephone, here's another good one. If you regularly advertise your phone number for the purpose of selling products or services, you are using your telephone for business purposes. In simple English, that means you must now have a business telephone instead of a residential telephone. Now a business telephone looks just like a residential telephone. It even looks the same way. All you do is pick up the receiver, wait for the dial tone, then dial the number. In fact, business telephones are exactly like residential telephones except for one small difference — they cost twice as much.

You're also going to have to give some thought to the government. They're always ready to stand by your side so they can collect their share of your hard-earned money. You will have to keep books, which means setting up an accounting system. Otherwise, you can't prove your expenses

and you may get taxed on all of your income even though most of it was spent on legitimate business expenses. While you're doing all of this book-keeping, YOU might as well help your state collect ITS sales tax.

I'm trying to make two points here. First, there are ways to market a program yourself with very little money if you're willing to compensate for it with a lot of extra work. But, if you do have money, you can save yourself some work and do a better marketing job. Second, there are a lot of things to be considered and to be worked out before you get started. Otherwise, you're going to feel that you just got yourself into something you wish you'd have never gotten involved with.

THE PUBLISHER

The word "publish" really means "to make public". A publisher, then, is one who makes things public. Normally, we associate publishers with books and magazines. However, since there is such a similarity between writing a computer program and writing a book or an article, a person who makes and sells computer programs submitted by others is also called a publisher. There are, however, differences between the software publisher and the publisher who works with the written word — but that's for another time.

The first step is to select a publishing company that would be interested in your product. Publishers tend to specialize. Some are good at marketing business-type software while others have established themselves in games.

After identifying a suitable publishing company, submit your program along with all the documentation you have. Whoever evaluates your program is not going to have a lot of time to figure things out. They're going to read your documentation first to see if it is a program that sounds interesting. If it is, they're going to set it up to run along with a lot of other programs to see how it works. It now becomes a question of how long you can keep their attention.

Assuming it's a good program, the publisher will contact you to give you the company's terms. The kind of terms you can expect to get is a subject I'll cover later on. For the moment, let me say this: Think about the offer for a few days and then respond. A good publishing company will never use your submission until you've agreed to its terms. If you don't like the terms, don't accept them. It's still your pro-

gram and you have the right to refuse any offer. You do, however, have an obligation to notify the company of your decision as soon as possible. If you do accept the terms, then your work is done. This is the beauty of having somebody else market the product for you.

The publishing company basically is going to deal with all of the problems I outlined earlier. It will keep the books, pay the taxes, and make the phone company happy. Because it is a business, it can take charge orders. Just about all publishers advertise in magazines so they have the means to promote your program. Finally, they are well-equipped to handle phone calls in a professional manner — answering questions and giving customers assistance.

“Not everybody likes to market through a publishing company. One reason is ignorance. There's a lot more to publishing than most people think.”

What you get is a check every month and a statement at the end of the year for your income tax records. No book-keeping is required on your part unless you want to deduct your own expenses for making the program that made the money. Talk with somebody knowledgeable about the tax laws on this. You may find some nice deductions with very little effort on your part.

Not everybody likes to market through a publishing company. One reason is ignorance. There's a lot more to it than most people think. Assuming you know what you're getting into, one very good reason to publish yourself is so you keep control of your product. Being in control, you can decide what the package will look like and what the ad copy will say. You can also decide how much advertising space should be devoted to your product in computer magazines. If you can handle it, that's great. Realistically, though, most people can't and a publisher will make the best decisions for all parties involved.

Another reason for publishing a program yourself might be that there isn't a publisher who can effectively market this type of program for you. For ex-

ample: If you wrote a program which generated a program log for a radio or television station, you'd have a hard time finding a publisher who reaches this market well. Not only that, there are almost no publishers who have any experience in this field and can take on the customer support when questions come up.

One new trend in software publishing works this way: Let's say that I decide to publish a game myself but I don't have the facilities to manufacture the tapes and to package them. What I might now do is go to a company like Adventureland and have them manufacture and package the product for me to my specifications. The rest of the marketing operation is still all mine. This way I have total control over my product, but I don't get hung up over the manufacturing part of the marketing operation.

You can also turn this around. Instead of using a publisher to do all of the work, you can do the manufacturing yourself. You then sell the finished product to the large publishers and they, in turn, just distribute and advertise it.

CHOOSING A PUBLISHER

Sometime after I lost my interest in computers and found my interest in photography again, I was very active in photography. I was never famous, but I managed to achieve what I consider to be a small amount of success.

One afternoon, a young woman who worked with me and who was starting out in photography, showed me some pictures she was planning to exhibit and to try to sell. She asked for my opinion.

I took one look at her pictures and immediately pronounced them to be terrible. She wanted to know why and I told her I didn't know. Then she wanted to know what made me think they were terrible. I told her the reason was that the photographer who took the pictures thought they were terrible. Well, she didn't think they were terrible and they were her pictures!!!

Now we get down to the heart of the matter. I pointed out to her that the prints looked like they were done by one of those \$1.25 super discount photo labs that advertise their specials every morning at three o'clock during the WPIX late movie. The frame was a cheap cardboard mat which obviously cost \$1.00 at a department store. In short, it looked like the photographer

continued on next page

continued from previous page

didn't want to spend any money on the pictures because they weren't worth anything.

Then I told her to think about my own pictures which she had seen. They were all hand enlarged, carefully printed, custom-cropped, color balanced exactly, dust-spotted, given a protective spray coating, and attractively framed. While you may not like my pictures, at least you would know from looking at them that I liked them. After all, if I don't like my own pictures, why should anybody else?

Since our discussion, the woman quit her job, got married, had a baby, and has never asked for my advice again.

Really, all I'm telling you is something about human nature. We like people who like themselves and dislike people who are at odds with themselves. But it's not just people. It's anything. Some of the worse programs I've seen are the ones that gave me the impression that the author was telling me, "I gave it a try and this is what I came up with." Gee, if the author isn't sure that program is any good, why should I be sure? On the other hand, if you think about some of the best programs you've ever bought,

didn't you get the feeling that whoever wrote the program was proud of it? Maybe you never thought about it, but it's true. Good programs always convey this impression.

The first impression we get of anything is from the way it looks. That's why the packaging is so important. It's got to convey the feeling of pride. It should never stop there. Once the purchase is made, the user should continue to feel that same feeling of pride long after starting to use your program.

But let me take this one step further. If a publisher produces a product that is packaged with pride, then that tells you, as a programmer, something about the publisher. Certainly, if you're going to let a company publish your work, you'll want one who will treat your program like it was something important and something it's proud to sell. Your first clue is the way the publishing company puts it all together — the appearance of the products it sells. So, the game becomes one where the author projects the image of pride in the work and the publisher helps to convey that impression from the beginning with the package.

When you find a publisher like this, all of the other considerations will

almost always fall into place. Such publishers will give you the best royalty arrangements you can get. Then, when they start selling your program, they will service the product.

By servicing the product, I refer to two things. First, the quality control that goes into the product. Second, customer support after the sale is made. Remember, if a tape doesn't load, it's YOUR PROGRAM that doesn't load. If the user can't figure something out and get help, it's YOUR PROGRAM that doesn't work.

From time to time, people come to me looking for jobs. This business about attitude is one thing I always bring up. I can teach anybody how to run a computer. I can teach anybody how to program a computer. But I don't know how to teach people how to put pride in their work and produce something that says, "The author was proud of this." The secret is all in your attitude and it's something that can't be taught.

FOOT NOTE: Just before sending this article to **SoftSide**, I found out that there was a Craps game program which was sold by G/2 — now out of business. It apparently played a full game of Craps with all the betting options.

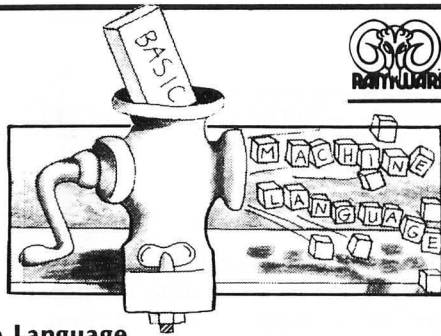


MACHINE HEAD

BY SPYDER



Tiny Comp



The Lazy Man's Shortcut to Machine Language

A BASIC Compiler in BASIC! Run your source program in BASIC, compile it into FAST Z-80 Code and execute the compiled version — all without reloading. 26 integer variables, GOTO, GOSUB, END, REM, RND, LET, +, *, /, IF, THEN, =, INKEY\$, CLS, PRINT@, CHR\$, PEEK, POKE. Compiled programs may be saved via TAPEDISK.

Supplied with game program, "3-D TIC TAC TOE", which uses all of the TINY COMP statement set and is ready to compile.

Manual includes several sample programs as well as thorough documentation of the Compiler for those who like to know "how things work" and for those who might even wish to EXPAND on TINY COMP's capabilities.

16K, S-80 Cassette \$19.95
32K, S-80, Disk \$24.95



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 certificates extolling their virtues as programming wizards, for all the world to see!

Send submissions to:

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



MOVE 1-2-3

MOVE 3-2-1

Move 1-2-3 is a brand new concept in computing technology. Now you can convert from Mod. I to Mod. III by file name or extension without having to convert an entire diskette. You may also select from a list as it passes in review. And with Move 3-2-1 you can convert back again!

Move 1-2-3 requires: Mod. III, Two Drives and 32k.

Move 3-2-1 requires: Any DOS, Mod. I, Two Drives, 32K and the PERCOM DOUBLER.

Convert Yourself for only \$29.95 #26-275001D



6 South St., Milford, NH 03055

(603) 673-5144

TOLL FREE OUT-OF-STATE

1-800-258-1790



The Sensuous Programmer

by "J"

The Fourth Enticing Entry

One of the nice things about computers is that you can tell them where to go, under exactly what circumstances, and they obey without a moment's hesitation. How many PEOPLE do you know who are so obliging?

There is, of course, a built-in logical flow to every computer program. That flow is primarily controlled by the program's line numbers. It makes no difference how randomly you type in the various instructions of a computer program; the computer looks not at the order in which they are entered, but at the order of their line numbers. So insistent is the infernal machine on having its way in this matter, that no matter how you try to confuse it, it will always list the program lines in ascending numerical order.

This comes as no great shock to most of you, I'm sure. (I do, though, remember this as one of the fascinating things about my very first hands-on contact with a computer — along with seeing letters appear on my TV screen, and watching a screen full of text scroll upward after the last line was filled!) The line numbers can jump by ones or by ten-thousands or whatever increment you choose, but the flow of the program will always hop from one line to the next higher line.

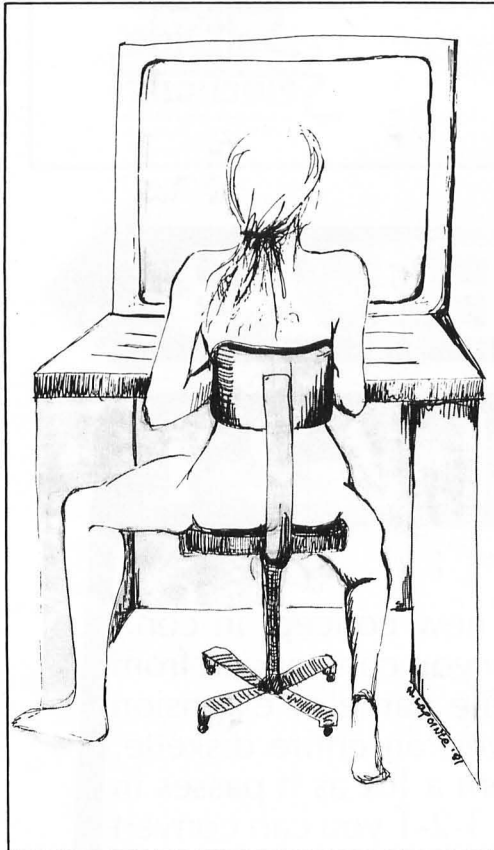
Unless, of course, you take the trouble of telling the computer where to go. You can make a program run backwards if you want to badly enough. Like this, for example:

```
10 GOTO 50
20 END
30 GOTO 20
40 GOTO 30
50 GOTO 40
```

It doesn't QUITE run backwards, since it does insist upon starting at the lowest line number, but you get the idea. The flow is virtually reversed, going from 10 to 50 to 40 to 30 to 20. (We could force the ultimate reversal to happen by dropping line 10 and giving the command "RUN 50" in place of the usual "RUN".)

Such a program, of course, doesn't do anything useful except demonstrate a point. But there are countless in-

stances where something very useful can be done by overriding the normal flow of a program from lowest to highest line numbers. You may have some doubts about the "usefulness" of the following program, but at least it's



a little less trivial than the previous one:

```
10 PRINT "HI, I'M ATAPPLE-80;
WHAT'S YOUR NAME?"
20 INPUT N$
30 PRINT "PLEASED TO MEET
YOU, ";N$
40 PRINT "WHO'S THAT STAND
ING NEXT TO YOU?"
50 GOTO 20
```

Obviously, if it's repetitive, inane conversation that you want, this program will give it to you. A whole evening's entertainment for a group of 500 is pretty good for a five-line program, no? And it's all made possible by that tricky little line 50, which gives you ab-

solute power over the flow of the program, overriding all the computer's in-born instincts to jump to the next highest line number.

Ah, but what if such absolute power is a little too heady and autocratic for your tastes, and you want to give the computer a bit of a role to play in deciding where to go next? In addition to its willingness to go where you want it to go as it executes a set of instructions, one of the most useful features of a computer is its ability to make certain kinds of decisions. You, as the programmer, must precisely define the type of decision to be made; but once you do that, the computer will unerringly make the "right" choice and act accordingly.

The most common decision-making instruction available to the BASIC programmer is the "IF...THEN" statement. "GOTO" allows you to tell the computer unequivocally where to go (that's called "branching"). "IF...THEN" leaves the decision up to the machine, according to the decision-making criteria you've given to it (that's called "conditional branching"). Here's an example of branching, both conditional and unconditional: a simple program that tells you whether a number that you type in is positive or negative.

```
10 PRINT "TYPE IN ANY
NUMBER: ";
20 INPUT N
30 IF N < 0 THEN GOTO 60
40 PRINT "YOUR NUMBER IS
POSITIVE."
50 GOTO 70
60 PRINT "YOUR NUMBER IS
NEGATIVE."
70 END
```

The one thing about having a computer make decisions for you is that you have to program it to consider all the possibilities. If you miss one, there's no way that dumb machine is ever going to come up with the smarts to help you out. The above program is a good example of a logical gap between the programmer's ears. It works — but not quite all the time. The set of instructions (the program) gives the

computer the ability to distinguish between positive and negative numbers, which certainly covers most of the territory. But it leaves out the unique case of the non-negative, non-positive number, zero. And because the programmer didn't think of that, the computer will blow it if somebody types in a zero. Line 20 checks to see if N is less than zero. Zero, of course, is NOT less than zero. So the test fails, the program does not branch to line 60, and it continues in the normal flow pattern of going to the next higher line number. That happens to be line 40, which prints the message "YOUR NUMBER IS POSITIVE.", and the ignorance of the machine once again outmaneuvers the intelligence of man.

In order to deal with all the possibilities, the programmer's instructions to the computer on making the decision must be altered:

```
10 PRINT "TYPE IN ANY
NUMBER: ";
20 INPUT N
30 IF N < 0 THEN GOTO 60
35 IF N = 0 THEN GOTO 65
40 PRINT "YOUR NUMBER IS
POSITIVE."
50 GOTO 70
60 PRINT "YOUR NUMBER IS
NEGATIVE."
62 GOTO 70
65 PRINT "YOUR NUMBER IS
ZERO."
70 END
```

This program will now be able to cope with every possible real number within the number-handling capabilities of the computer.

The "IF...THEN" statement is even more versatile than the above example suggests. Since any valid BASIC statement can follow the "THEN", the above program could be made quite a bit neater and easier to follow by rewriting it in this way:

```
10 PRINT "TYPE IN ANY
NUMBER: ";
20 INPUT N
30 IF N > 0 THEN PRINT "YOUR
NUMBER IS POSITIVE."
40 IF N < 0 THEN PRINT "YOUR
NUMBER IS NEGATIVE."
50 IF N = 0 THEN PRINT "YOUR
NUMBER IS ZERO."
60 END
```

Since only one of the three lines 30-50 can be true, only one of the

PRINTs will be executed and the rest skipped.

Or, yet another way of coding the program would be like this:

```
10 PRINT "TYPE IN ANY
NUMBER: ";
20 INPUT N
30 IF N > 0 THEN PRINT "YOUR
NUMBER IS POSITIVE.": GOTO
60
40 IF N < 0 THEN PRINT "YOUR
NUMBER IS NEGATIVE.": GOTO
60
50 PRINT "YOUR NUMBER IS
ZERO."
60 END
```

This approach trades in one conditional branch for two unconditional ones (if that's your idea of a good bargain), illustrates the use of multiple statements on a line, and is likely to increase execution speed slightly. In most BASICs, when the condition checked by an "IF" is false, the program will continue at the next higher line, as mentioned above (Apple's Integer BASIC is an exception). This means that by judicious use of the colon (:), you can get the computer to do not just one but several things if the condition you're testing is true — skipping over all those things if the condition is false. This is not always the most desirable approach, but it comes in handy quite frequently, as in the above program.

As usual, there are also other ways to skin a cat. (No offense intended to our feline readers.) Conditional branching can be accomplished through at least two other types of BASIC structures: the computed GOTO and the "ON...GOTO" statement (plus the corresponding GOSUBs). The Atari gives you both of these; the Apple gives you the computed GOTO in Integer and "ON...GOTO" in Applesoft; and the S-80 gives you "ON...GOTO".

The computed GOTO allows you to branch to a line number which is determined by the value of a variable or a formula. Here's a way to rewrite the number-evaluation program above using the computed GOTO along with BASIC's SGN function (which yields a +1, 0, or -1 depending on the sign of the number):

```
10 PRINT "TYPE IN ANY
NUMBER: ";
```

```
20 INPUT N
30 S = SGN(N)
40 GOTO 100 + 10 * S
90 PRINT "YOUR NUMBER IS
NEGATIVE.": GOTO 120
100 PRINT "YOUR NUMBER IS
ZERO.": GOTO 120
110 PRINT "YOUR NUMBER IS
POSITIVE."
120 END
```

The formula $(100 + 10 * S)$ will yield a result of 90, 100, or 110 corresponding to values of -1, 0, and 1 for S; so those are the line numbers to which control can be passed by this computed GOTO instruction.

One of the nice things about this kind of structure is being able to name various parts of your program (especially subroutines) by their line number. This is a form of self-documentation which can really help to make the flow of a program clear. For instance:

```
10 BEGIN = 1000: GAME = 2000:
ENDGAME = 3000
20 GOSUB BEGIN
30 GOSUB GAME
40 GOTO ENDGAME
1000 (etc. . .)
```

The "ON...GOTO" statement, also, is useful for multiple-branching arrangements. Reprogramming the number evaluator again, it might look like this:

```
10 PRINT "TYPE IN ANY
NUMBER: ";
20 INPUT N
30 S = SGN(N) + 2
40 ON S GOTO 90, 100, 110
90 PRINT "YOUR NUMBER IS
NEGATIVE.": GOTO 120
100 PRINT "YOUR NUMBER IS
ZERO.": GOTO 120
110 PRINT "YOUR NUMBER IS
POSITIVE."
120 END
```

You can list as many line numbers following the GOTO (or GOSUB) as you can fit into a program line, and the control will branch to the first, second, third, etc., corresponding to the value of the variable named (S in this case).

But what if the variable has some weird value, such as 4203 or -6.95?

continued on page 23

MURA HI STEPPER

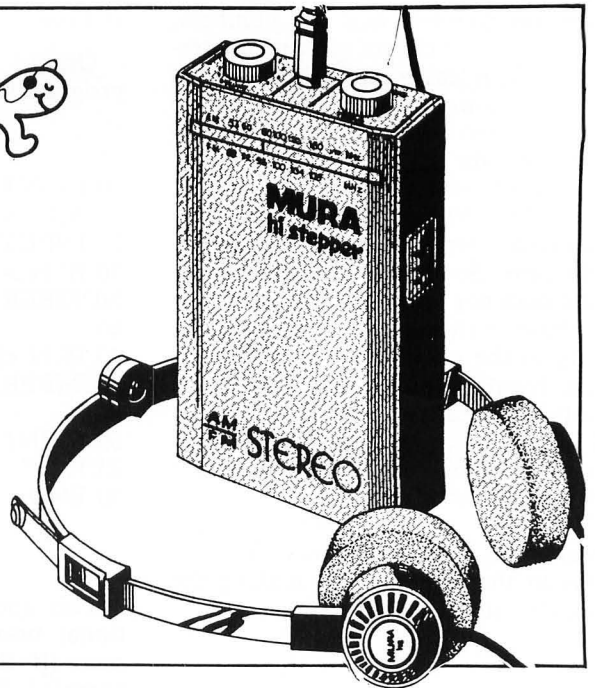
Mura's "HI STEPPER" is a pocket size AM/FM/FM Stereo Radio designed for people on the move. "HI STEPPER" features include ● Automatic and manual stereo switching ● Slide rule tuning ● Stereo balance control ● LED stereo indicator ● Powered by 3 AA batteries (that provide 10 times longer playing time than a 9V battery) or with DC adaptor ● Built-in pocket/belt clip ● Weighs only 9.1 oz. including batteries ● Measures 1.2 x 3.1 x 4.7 in. ● Carrying case included. The "HI STEPPER" is finished in a handsome ebony high gloss.

The ultra lightweight headset (1.6 oz., less cord) is Mura's model hs. It has ● Rare earth (samarium cobalt) drivers ● Tapered mylar speakers ● A 4 ft. lightweight cord with a 3.5mm plug. This has also features a foldable headband that permits compact storage and convenient carrying when not in use.

#99-05 \$59.95

(plus \$2.50 shipping and handling)

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



BUGS, WORMS,

and other undesirables

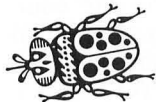
A bug has been reported in "International Bridge Contractors" (April, 1981, page 22). In line 255 the variables BS(Z) and BM(Z) should be set to 1000 and 1500, rather than 100 and 150 as given. These variables represent the single-beam bridge's safe and maximum lengths.

"Math Decathlon," part 3 (May, 1981) has an error in line 6070. The line should read:

```
6070 C = INT (RND(1) * 3 * (S(P) ^ 2 +
1)) + 1: IF C = B THEN 6070
```

In "Bats" (June, 1981), when you win or lose the game, the computer displays the number of bats in the next round rather than in the current round. To correct this the following changes should be made:

```
3010 B1 = B
4010 B1 = B
5005 PRINT 3960, "NO. BATS: "; B1;
```



In both the Apple and Atari versions of "Kidnapped" published in the July SoftSide, the following lines should be changed:

Atari

```
1580 IF A<>16 OR DP<>0 OR D$(1,3)<>"SLE" THEN 1590
2200 K3=56:R$="LIT FLASHLIGHT":GOSUB 2500:H$(55*SA+1,56*SA)="LIT FL
ASHLIGHT      ":DK=1:DT=0:GOTO2880
```

Apple

```
855 IF A<13 AND E$ = "GHT" AND I(56) = 1 THEN K3 = 56:R$ = "LIT FLA
SHLIGHT": GOSUB 1100:H$(56) = R$:DK = 1:DT = 0: GOTO 4900
```

★★ REWARD! ★★ TRANSLATION APPEAL

We will give away a \$100 software certificate each month for the best translation of a feature program in **SoftSide** magazine. Furthermore, we will publish the translation in the magazine. Your portfolio will be enhanced and you will garner fame and fortune for your efforts!

We will allow three months after initial publication of a program for the translation to be sent. After that time we will not accept entries. The quality of the translation will be judged by the **SoftSide** editorial staff and the winning entry will be published the following month, i.e., four months after publication of the original program.

Entries must be submitted on cassette or disk, accompanied by documentation. Please enclose a self-addressed stamped envelope if you would like your entry returned to you.



The Sensuous Programmer

continued from page 21

Well, negative values are bad news: your program will crash with an insulting error message. Non-integer values will have only their whole-number component evaluated: 2.78 would be treated as a 2, for example. Values which evaluate as 0 (0.999999999), or as a whole number greater than the number of lines listed, will cause control to "drop through" to the next higher line number, ignoring all the nice choices you've provided. The moral, of course, is that you ought to anticipate all possible values of the variable you use in an "ON...GOTO" or "ON...GOSUB", to eliminate unpleasant surprises. For example:

```
360 PRINT "PLEASE ENTER
YOUR CHOICE (1, 2, OR 3):";
370 INPUT C
380 IF C < 0 THEN GOTO 360
390 ON C GOTO 500, 600, 700
400 GOTO 360
500 (etc. . .)
```

Line 380 traps an improper response (a negative number) which would cause the program to crash, and line 400 "catches" all other responses that are either less than 1, or equal to or greater than 4. There are other approaches to trapping such errors (and this one can't cope very well with a non-numerical input), but this illustrates one possibility.

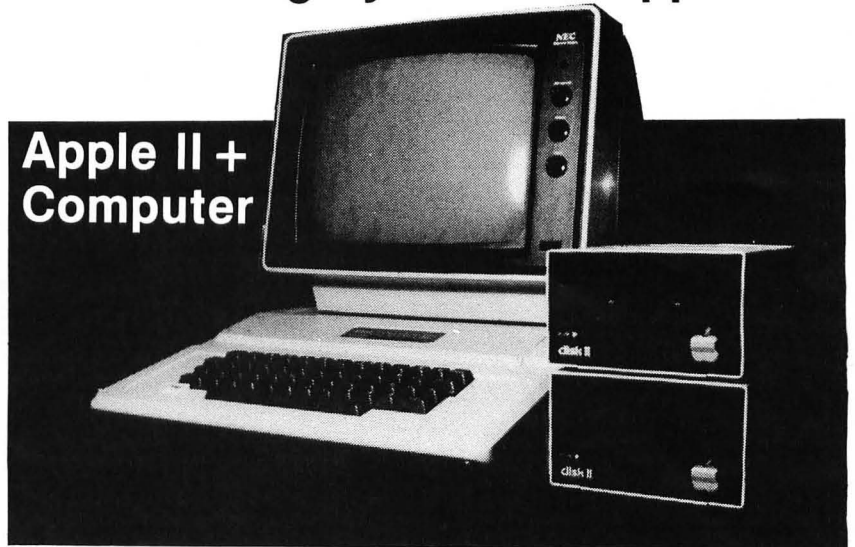
So much for branches, conditional and otherwise. Now you know how to tell your computer where to go, and how to give it decision-making power. With just a little freebie thrown in on trapping bad input. Next month, more on this last topic, with various hints and techniques designed to anticipate the unanticipated, can the uncanny, fool the foolish, and wittily outwit the half-wit.

Atari One Liner

```
1 CLOSE #1:OPEN #1,4,0,"K":GET #1,N:
FOR V=0 TO 3:SOUND V,N,10,1:NEXT V:GR
APHICS 2+16:SETCOLOR 4,N,N:PLOT RND(0
)14,RND(0)10:2 #6;"ORGAN":GOTO 1
```

Jonathan Schiff
Pasadena, CA

Take a big byte of the Apple...



**48K RAM
(#47-203).....\$1299.00**

...and keep nibbling at the edges with these peripherals, available from TSE HARD SIDE...

SUP-R-MOD RF Modulator (#47-100)	\$34.95
APPLE II Disk Controller w/Drive (#47-004).....	\$619.00
APPLE II Disk Drive (#47-005)	\$499.00
LOBO Disk Drive (#47-3101)	\$379.00
MICROSOFT Z-80 SoftCard (#47-80)	\$269.00
MICROSOFT RAMCard (#47-81)	\$159.00
Integer BASIC Language Card (#47-1C).....	\$195.00
PASCAL Language Card (#47-PAS)	\$459.00
MOUNTAIN COMPUTER Apple Clock (#47-MH003).....	\$269.00
MOUNTAIN COMPUTER ROMWriter (#47-MH015)	\$169.00
MOUNTAIN COMPUTER ROMPlus w/Filter (#47-MH007)	\$189.00
MOUNTAIN COMPUTER Music System (#47-MH022)	\$519.00
MOUNTAIN COMPUTER A/D + D/A (#47-MH023)	\$329.00
MOUNTAIN COMPUTER Expansion Chasis (#47-MH024).....	\$609.00
HARDSIDE Memory Upgrade Kit (#5-1102).....	\$39.00
APPLE Silentype Printer w/Interface (#47-000)	\$569.00
Parallel Printer Card and Cable (#47-9)	\$100.00
CCS Arithmetic Processor (#47-7811C).....	\$399.95
CCS Asynchronous Serial Interface (#47-7710A).....	\$159.95
CCS Synchronous Serial Interface (#47-7721A)	\$159.95
CCS GPIB (IEEE-488) Interface (#47-7490A)	\$300.00
CCS Calendar/Clock Module (#47-7424A)	\$125.00
LYNX Communications Interface (#19-85)	\$229.00
KURTA APPLE Graphics Tablet (#47-1000)	\$659.00
VERSAWRITER (#47-1100)	\$239.00
ALF AM-II Music Synthesizer (#47-1200)	\$189.00
LEEDEX 12" B/W Monitor (#5-100)	\$159.00
NEC 12" Monochrome Monitor (#5-200)	\$239.00
Color Video Receiver (#26-3010)	\$389.00

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



Quest 1

by Brian Reynolds

Apple translation by Rich Bouchard.
Atari translation by Alan J. Zett.

“Quest 1” is a graphics dungeon game which will run in 16K RAM on the S-80, or 24K RAM on the Apple or Atari.

In “Quest 1” you become a strong warrior who journeys through an ancient maze in search of four huge sapphires and other treasures. These precious jewels are guarded by terrible Wraiths, Giants, Mummies, and other unpleasant monsters. To find the treasures, then, you must be very strong (to kill the monster), very dextrous (to sneak around the monster and steal the treasure), or very intelligent (to list the program and cheat!).

When you begin your quest, a character will be created for you. He (or she) will be either an Elf, a Dwarf, or a Human. He will be given ratings in strength (3-20) and dexterity (3-20), and a percentage rating according to his wounds (100% = no wounds, 0% = a dead fighter). Being new to the field of questing, your warrior will not be much favored by the gods and will not have much magic to use. He will, however, have four different ways to fight: He will be given a random number of normal arrows, magic arrows, and holy water, plus his trusty sword. Some healing potions will also be given for restoring wounds.

After you have named your fighter, you will be teleported into a marketplace in a nearby town to bargain with a greedy merchant for more supplies. This usually takes only a short time, since the merchant will probably either sell to you quickly, or else refuse to sell at any kind of affordable price. After completing your bargaining, enter a ‘0’ to begin your quest.

When you enter the dungeon, a text-graphics display will be created on the screen, showing all your statistics in the corners and a picture of your current location in the center. If you have an S-80 Model III, your character will look different depending on his/her gender. On the Model I and on the Apple, both male and female characters are represented by an ‘@’ symbol. And in the Atari version, a specially-defined text character is used. (Watch for an upcoming article on creating such text

continued on next page



continued from previous page

characters yourself.) Treasure chests appear as asterisks (*), while monsters are shown by the initial letter of their name.

You can attempt your quest through the 58 rooms of the dungeon simply by killing monsters, taking the treasures, and moving on. However, this is not advisable for two reasons. First, you must remember the way out of the dungeon, or you will surely perish. And second, wandering monsters abound in this dungeon; if a wraith, for example, comes up behind you, he will probably kill you with one good blow. You should also be aware that frequent trips back for supplies are not wise, since more monsters are added each time you re-enter the dungeon with more than 100 experience points.

Note that the greater your dexterity rating, the more SLOWLY the game will seem to move. This is because your higher dexterity, in effect, gives you more time to think and react relative to the pace of the game. As you accumulate experience points, however, the pace and difficulty of the game will increase.

When you find your way out of the dungeon, the computer will give you a list of all the treasures you retrieved, add in any arrows or potions you may have found, give you a chance to save the game, and let you quit if you want to. If you do quit, the computer will give you a list of all your fighter's abilities and possessions so that you can use him in a later game. If you elect to continue, you are teleported back into the marketplace to get more supplies and then to continue your quest.

Commands are entered with single keystrokes, as follows:

S-80 version

The number keys are convenient if you have a numeric keypad:

- 8 or up-arrow: Move up.
- 4 or left-arrow: Move left.
- 6 or right-arrow: Move right.
- 2 or down-arrow: Move down.

Apple and Atari versions

The keys form a diamond shape:

- W: Move up.
- A: Move left.
- D: Move right.
- X: Move down.

All versions:

Any key other than above: Stop movement.

N: Shoot a normal arrow (not effective against Wraiths).

M: Shoot a magic arrow.

T: Toss a vial of holy water (affects only "undead" monsters: Skeletons, Zombies, Ghouls, Mummies).

F: Fight in close combat (not effective against Giants or Wraiths).

O: Open a treasure chest when you are next to it. (It will disappear and its contents will be displayed on the screen.)

H: Drink a healing potion. (This restores your wound rating to 100%.)

Below is a complete inventory of the monsters, with their wound ratings. These ratings represent the monster's strength, relative to your initial strength. If you are attacked, by a skeleton for example, it can inflict wounds of up to 20% on you with each hit. And, it takes more to kill a monster with a high rating than one with a low rating.

- Skeleton: 20%
- Orc: 30%
- Zombie: 40%
- Ghoul: 50%
- Spider: 70%
- Mummy: 80%
- Giant: 90%
- Wraith: 99%

VARIABLES:

- A1: Number of normal arrows.
- A2: Number of magic arrows.
- DX: Dexterity rating.
- EP(*): Experience value of each treasure.
- GP(*): Gold value of each treasure.
- HW: Number of vials of holy water.
- MS(*): Single-character monster identifier.
- M1(*): Type of monster in each room.
- M2(*): Number of monsters in each room.
- MN\$(*): Names of the monsters.
- MS(*): Standard wound value for each monster.
- NM\$: Name of fighter character.
- OP: Original price of an item at the market.
- P1: Current price of an item at the market.
- PT: Number of healing potions.
- R1(*): Identifies each location as either a passage/intersection (= 1) or a chamber/room (= 2).
- R2(*,*) : For each room, identifies what room you will enter by exiting up, down, left, and right respectively.
- RC: Race of fighter (0 = Human, 1 = Elf, 2 = Dwarf).
- RM: Current room number.
- ST: Player strength.
- T\$(*) : Name of each treasure.
- T1(*): Identifies type of treasure in each room.
- TRS80MODEL: Model number of computer (S-80 version).
- TS(*): Quantity of each treasure type retrieved by player.
- TX,TY: X and Y coordinates of treasure.
- W: Wounds (multiply by 100 to get percentage).
- WX,WY: X and Y coordinates of monster.
- X5,Y5: X and Y coordinates of player.
- YY\$: Single-character identifier for player.

S-80 VERSION

Print title page.

```

1 CLS: CLEAR400: IFPEEK(664)=58 AND PEEK(665)=16 THEN TRS80MODEL=3 ELSE
  TRS80MODEL=1
2 PRINT@476,"QUEST 1";: IF TRS80MODEL=3 THEN POKE16420,1
3 A$="          * * * * *   Q U E S T   1   W
  A S   W R I T T E N   B Y   B R I A N   R E Y N O L D S   * * *
  * * * * *   ( H A V E   F U N ! )   "
4 FORX=1 TO LEN(A$)-14: PRINT@537,MID$(A$,X,14);: FORY=1 TO 30: NEXT: NEXT:
  XT: PRINT
5 ON ERROR GOTO 30000
6 FORX=1 TO 1000: NEXT

```

7 PRINT

```

8 IF TRS80MODEL=1 THEN Y$="@" ELSE IF TRS80MODEL=3 THEN Y$=CHR$(253)

```

Data for monsters and treasures.

```

100 DATA "WORTHLESS ODDS & ENDS",0,0,"A BAG FULL OF COPPER COINS",1,3,
  "A SMALL BRASS STATUETTE",2,5,"A BAG FULL OF VARIOUS COINS",3,7,
  "A PURSE FULL OF GOLD COINS",5,12,"3 GOLD NUGGETS",8,17,
  "4 SMALL TURQUOISES",7,15,"LARGE RUBY",15,30
105 DATA "A HUGE SAPPHIRE",150,150,"A HEALING POTION",10,0,"A
  QUIVER OF 10 MAGIC ARROWS",15,0,"A QUIVER OF 10 NORMAL ARROWS",1,0,0

```

```
110 DATA "SKELETON", "S", 2, "ORC", "O", 3, "ZOMBIE", "Z", 4, "GHOUL", "G",  
6, "HUGE SPIDER", "H", 7, "MUMMY", "M", 8, "GIANT", "G", 9, "WRAITH", "W", 9  
.9
```

Data for the rooms.

```
115 DATA 1,12,3,2,18,0,0,0  
120 DATA 2,0,0,0,1,4,2,8  
125 DATA 1,1,0,4,19,0,0,1  
130 DATA 1,0,0,5,3,3,1,1  
135 DATA 2,6,38,0,4,1,3,6  
140 DATA 1,8,5,9,7,0,0,0  
145 DATA 1,0,0,6,0,0,0,1  
150 DATA 2,0,6,0,11,2,11,2  
155 DATA 2,0,0,10,6,2,3,1  
160 DATA 2,0,0,0,9,5,1,4  
165 DATA 1,0,0,8,12,0,0,1  
170 DATA 2,0,1,11,13,2,5,3  
175 DATA 1,0,0,12,14,0,0,1  
180 DATA 2,15,26,13,17,5,1,1  
185 DATA 2,0,14,0,0,0,1  
190 DATA 2,0,17,0,0,1,2,5  
195 DATA 1,16,20,14,0,4,1,1  
200 DATA 2,0,19,1,26,2,2,7  
205 DATA 2,18,30,3,27,3,2,2  
210 DATA 1,17,21,0,0,0,0,1  
215 DATA 1,20,22,0,0,6,2,9  
220 DATA 1,21,23,0,0,2,3,12  
225 DATA 1,22,24,0,0,4,2,10  
240 DATA 1,23,25,34,0,0,0,11  
250 DATA 2,24,0,0,0,7,3,9  
260 DATA 2,14,0,18,0,3,2,1  
270 DATA 2,0,28,19,0,4,1,2  
280 DATA 1,27,29,31,0,0,0,1  
290 DATA 2,28,0,0,0,5,1,10  
300 DATA 2,19,0,0,0,1,2,3  
310 DATA 1,0,32,0,28,0,0,4  
320 DATA 1,31,33,43,0,0,0,1  
330 DATA 2,32,35,0,0,5,1,8  
340 DATA 1,0,0,35,24,0,0,12  
350 DATA 1,33,36,45,34,0,0,5  
360 DATA 1,35,0,37,0,7,1,10  
370 DATA 2,0,0,0,36,8,3,9  
380 DATA 1,5,49,0,39,0,0,1  
390 DATA 1,0,40,38,0,0,0,6  
400 DATA 1,39,0,0,41,2,3,2  
410 DATA 1,42,46,40,43,4,1,7  
420 DATA 2,0,41,0,0,7,3,8  
430 DATA 2,0,44,41,32,6,1,11  
440 DATA 1,43,45,0,0,0,0,5  
450 DATA 1,44,0,47,35,0,0,1  
460 DATA 2,41,47,48,0,5,1,7  
470 DATA 1,46,0,50,45,0,0,3  
480 DATA 1,0,0,49,46,0,0,1  
490 DATA 2,38,51,52,48,6,1,6  
500 DATA 1,0,0,51,47,2,5,10  
510 DATA 1,49,0,53,50,4,3,5  
520 DATA 2,0,0,0,49,6,1,6  
530 DATA 2,0,54,0,51,5,1,8  
540 DATA 1,53,0,0,55,0,0,1  
550 DATA 2,0,0,54,56,2,3,2  
560 DATA 1,0,0,55,57,6,1,8  
570 DATA 1,0,0,56,58,7,3,11  
580 DATA 2,0,0,57,0,8,4,9
```

Initialize variables.

```
600 DIM MN$(8),M$(8),MS(8),R1(58),R2(58,4),M1(58),M2(58),T1(58),
```

```
T$(12),EP(12),GP(12)  
603 FORX=1TO12:READT$(X),EP(X),GP(X):NEXTX  
605 FORX=1TO8:READMN$(X),M$(X),MS(X):NEXT  
610 FORX=1TO58:READR1(X):FORY=1TO4:READR2(X,Y):NEXTY  
615 READM1(X),M2(X),T1(X):NEXTX  
620 RM=1:A1=1000:A2=1000:W=1:PT=2  
625 IFTRSBOMODEL=3THENPOKE16409,1
```

Use an old character?

```
800 IFB1=1THENGOSUB20000  
805 IFB1=1THENB1=0:GOTO900  
810 INPUT "DO YOU WISH TO USE AN OLD CHARACTER";A$:IFLEFT$(A$,1)  
<>"Y"THENGOSUB21000:GOTO900  
811 IFTRSBOMODEL=3THENPOKE16409,0  
812 INPUT "NAME ";NM$  
813 IFTRSBOMODEL=3THENPOKE16409,1  
815 INPUT "STRENGTH ";ST:IFST>20ORST<3THENB15  
820 INPUT "DEXTERITY ";DX:IFDX>20ORDX<3THENB20  
825 INPUT "WOUNDS ";W:W=W/100:IFW<.1ORW>1THENB25  
830 INPUT "EXPERIENCE ";EP:INPUT "GOLD ";GP  
835 INPUT "IS (S)HE AN ELF";A$:IFLEFT$(A$,1)="Y"THENRC=1  
836 IFRC=0THENINPUT "IS (S)HE A DWARF";A$:IFLEFT$(A$,1)="N"THENRC  
=2  
840 INPUT "MAGIC ARROWS ";A2:INPUT "NORMAL ARROWS ";A1  
845 INPUT "HEALING POTIONS ";PT  
846 INPUT "HOLY WATER ";HW  
847 IFTRSBOMODEL=3THENINPUT "Is this character female";A$:IFLEFT$(  
A$,1)="Y"THENYY$=CHR$(254)
```

Load in an old game?

```
850 INPUT "Do you wish to load in an old game";A$:IFLEFT$(A$,1)<>  
"Y"THEN900  
860 INPUT "From cassette or disk ";A$:IF LEFT$(A$,1)="C"THENINPUT  
"PRESS ENTER TO BEGIN LOAD ";A$:GOTO880  
862 IF LEFT$(A$,1)<>"D"THENB60  
870 OPEN "I",1,"QUEST/DAT"  
872 FORX=1TO58:INPUT#1,M1(X),M2(X),T1(X):NEXT  
876 CLOSE:GOTO900  
880 FORX=1TO58:INPUT#-1,M1(X),M2(X),T1(X):NEXT
```

Marketplace and bargaining routine.

```
900 CLS:PRINT "Gold: ";GP  
901 PRINT "You are at the market. Prices here are:"  
902 PRINT  
903 PRINT "1. Magic Arrow -----  
-- 2 gold"  
904 PRINT "2. 4 Normal arrows -----  
-- 1 gold"  
905 PRINT "3. Healing potion -----  
- 15 gold"  
906 PRINT "4. Holy water -----  
-- 3 gold"  
910 PRINT " Ok, ";NM$; ", what item would you like (number)";:INPU  
TIT:IFIT>4ORIT<0PRINT "I don't sell THAT.":GOTO910  
911 IFIT=0THEN990  
912 IFIT=1THENP1=2ELSEIFIT=2THENP1=1ELSEIFIT=3THENP1=15ELSEIFIT=  
4THENP1=3  
915 PRINT " At ";P1;" gold apiece, how many will you buy";:INPUTN  
M:IFNM<1THENPRINT "Very funny. I do not BUY things, I SELL them."  
:GOTO915  
920 P1=P1*NM  
921 OP=P1  
925 PRINT "The price now comes to ";P1;" gold."  
930 PRINT "How much will you give me, ";NM$;:INPUTA
```

continued on next page

continued from previous page

```
935 IFA<OP/10>THENPRINT"FORGET IT!!!!":GOTO901
940 IFA<OP/2>THENPRINT"Not interested.":GOTO901
941 IFA>P1THENPRINT"You got a deal!!!!":GOTO950
942 Y=A/P1:X=RND(0):IFX>YTHENPRINT"Not interested":P1=INT((OP+P1)/2):GOTO930
945 P1=INT((P1#2+A)/3):PRINT"How about ";P1; ", ";NM$;"?":GOTO930
950 IFGP<P1THENPRINT"what!!! Can't pay yer debts??? You'll be th
rown in prison for this, ";NM$;"!!!":END
955 GP=GP-P1:PRINT"You now have ";GP;" gold, ";NM$;"."
960 IFIT=1THENA2=A2+NM
965 IFIT=2THENA1=A1+NM#4
970 IFIT=3THENPT=PT+NM
975 IFIT=4THENHW=HW+NM
980 GOTO901
```

Enter dungeon; check for too many arrows.

```
990 PRINT"Ok, ";NM$; ", Press <ENTER> to go into the dungeon."
991 EL=0
992 IFEP>100THENEP=EP-100:EL=EL+100:FORX=1TO58:M2(X)=M2(X)*1.1:N
EXT:GOTO992
993 EP=EP+EL
994 IFEL>500THENFORU=ELTO500STEP-100:FORX=1TO58:M2(X)=M2(X)/1.1:
NEXT:GOTO992
995 INPUTA$:CLS
997 A3=0:A4=0
998 IFA2>ST#2THENA4=A2-ST#2:A2=ST#2:PRINT"MORE THAN ";ST#2;" MAG
IC ARROWS WOULD WEIGH YOU DOWN.":FORX=1TO1000:NEXT
999 IFA1>ST#2THENA3=A1-ST#2:A1=ST#2:PRINT"MORE THAN ";ST#2;" ARR
OWS WOULD WEIGH YOU DOWN.":FORX=1TO1000:NEXT
```

Upon entering a new room, draw it with its monsters and treasures. If this is room one, give option to leave.

```
1000 CLS
1001 IFB1=0THENB1=1ELSEIFR1=1THENINPUT"Do you wish to leave the
dungeon";A$:IFLEFT$(A$,1)="Y"THENB00ELSECLS
1005 DNR1(RM)GOSUB10000,11000
1010 IFT1(RM)>0THENTX=RND(39-23)+23:TY=RND(4)+5:PRINT@TX+64*TY,"
#";
1015 X5=31:Y5=8
1020 IFI$="8"THENY5=14ELSEIFI$="2"THENY5=1ELSEIFI$="6"THENX5=1EL
SEIFI$="4"THENX5=62
1025 PRINT@X5+64*Y5,YY$;
1030 IFM2(RM)>1THENWX=RND(39-23)+23:WY=RND(4)+5
1031 MS=MS(M1(RM))/10
1050 IFM2(RM)>1THENPRINT@WX+64*WY,M$(M1(RM));
```

Print player status, check for wandering monsters.

```
1055 PRINT@0,"ARROWS: ";A1;:PRINT@192,"M. ARROWS: ";A2;
1060 PRINT@960,"ST=";ST;" DX=";DX;:PRINT@64,"WOUNDS: ";LEFT$(STR
$(W#100),6);"% ";
1061 PRINT@128,"ROOM: ";RM;
1062 PRINT@832,"HEALING POTIONS: ";PT;
1063 PRINT@896,"HOLY WATER: ";HW;
1065 IFM2(RM)>1THENPRINT@42,"MONSTER: ";MN$(M1(RM));
1070 IFM2(RM)<1THENPRINT@42,STRING$(64-42,128);
1075 IFM2(RM)>1THENPRINT@42+64,"NUMBER: ";INT(M2(RM));:ELSEPRINT
@42+64,STRING$(20,128);
1080 PRINT@704,"EX POINTS: ";INT(EP);
1085 PRINT@768,"GOLD: ";GP;
1086 IFM2(RM)=0ANDRND(100)=1THENFORX=1TO10:PRINT@42,"Wandering M
onster!";:FORY=1TO50:NEXT:PRINT@42," ";:FORY=
1TO50:NEXT:NEXT:M2(RM)=RND(3):M1(RM)=RND(8):GOTO1030
```

Accept a command from keyboard and call appropriate subroutines.

```
1090 FORX=1TO(DX#10)-EP:A$=INKEY$:IFA$=""THENNEXTELSEX=3550:NEXT
1093 IFT1(RM)>0THENPRINT@TX+64*TY,"#";
1095 IFA$<>""THENIFA$="["THENI$="8"ELSEIFA$=CHR$(10)THENI$="2"EL
SEIFA$=CHR$(9)THENI$="6"ELSEIFA$=CHR$(8)THENI$="4"ELSEI$=A$
1100 IFI$="8"THENGOSUB15100
1105 IFI$="2"THENGOSUB15200
1110 IFI$="6"THENGOSUB15300
1115 IFI$="4"THENGOSUB15400
1120 IFI$="H"THENI$="":IFPT>0THENPT=PT-1:W=1
1125 IFI$="M"ANDA2>0THENI$="":A2=A2-1:GOSUB15500
1130 IFI$="N"ANDA1>0THENI$="":A1=A1-1:GOSUB15600
1135 IFI$="F"THENGOSUB16000
1140 IFI$="O"THENGOSUB17000
1145 IFI$="T"ANDHW>0THENI$="":HW=HW-1:GOSUB 18000
```

If there is a monster in the room, move him and let him attack.

```
1200 IFM2(RM)<1THEN1030
1201 IFMS<0THENFORX=191TO128STEP-1:POKE15360+WX+(64*WY),X:NEXT:
M2(RM)=M2(RM)-1:EP=EP+MS(M1(RM)):GOTO 1030
1205 IFWX>X5THENMX=-1ELSEIFWX>X5THENMX=1ELSEMX=0
1210 IFWY>Y5THENMY=-1ELSEIFWY>Y5THENMY=1ELSEMY=0
1215 PRINT@WX+64*WY," ";
1220 IFPEEK(15360+WX+MX+64*WY)=128ORPEEK(15360+WX+MX+64*WY)=32TH
ENWX=WX+MX
1225 IFPEEK(15360+64*(WY+MY)+WX)=128ORPEEK(15360+64*(WY+MY)+WX)=
32THENWY=WY+MY
1230 IF(ABS(WX-X5)>1)OR(ABS(WY-Y5)>1)THEN1050
1235 X=RND(0):IFX>MSTHEN1050
1240 X=RND(0)*MS
1245 W=W-X:IFW<0THEN5000
1250 GOTO 1050
```

End-routine for the "Great Dungeon in the Sky" ending.

```
5000 FORX=191TO128STEP-1:POKE15360+X5+64*Y5,X:NEXT:FORX=1TO1000:
NEXT:CLS
5005 PRINT"WELCOME TO HEAVEN, ";NM$;"!!!"
5010 PRINT"I hope you enjoyed your short lifetime in which"
5015 PRINT"You accumulated ";GP;" gold and ";EP;" experience poi
nts."
5020 PRINT:PRINT:INPUT"Would you like to be reincarnated as a ne
w character";A$:IFLEFT$(A$,1)="N"THENENDELSERUN
```

Subroutine to draw a passage/intersection.

```
10000 REM
10005 X1=R2(RM,1)
10010 IFX1>0THENFORX=0TO320STEP64:PRINT@X+23,CHR$(191);:PRINT@X+
40,CHR$(191);:NEXTELSEPRINT@343,STRING$(18,188);
10015 X1=R2(RM,2)
10020 IFX1>0THENFORX=64TO960STEP64:PRINT@X+23,CHR$(191)+STRING$(
16,128)+CHR$(191);:NEXT:ELSEPRINT@663,STRING$(18,143);
10025 X1=R2(RM,3)
10030 IFX1>0THENFORX=41TO63:PRINT@320+X,CHR$(188);:PRINT@640+X,C
HR$(143);:NEXT:ELSEFORX=384TO576STEP64:PRINT@X+40,CHR$(191);:NEX
T
10035 X1=R2(RM,4)
10040 IFX1>0THENFORX=0TO22:PRINT@320+X,CHR$(188);:PRINT@640+X,CH
R$(143);:NEXT:ELSEFORX=384TO576STEP64:PRINT@X+23,CHR$(191);:NEX
T
10045 RETURN
```

Subroutine to draw a chamber/room.

```
11000 REM
11005 PRINT@192+16,CHR$(191)+STRING$(7,143)+STRING$(16,128)+STRI
NG$(7,143)+CHR$(191);
11010 PRINT@768+16,CHR$(191)+STRING$(7,188)+STRING$(16,128)+STRI
```

```

NG$(7,188)+CHR$(191);:FORX=192TO320STEP64:PRINT@X+16,CHR$(191);:
PRINT@X+47,CHR$(191);:NEXT:FORX=640TO768STEP64:PRINT@X+16,CHR$(1
91);:PRINT@X+47,CHR$(191);:NEXT
11015 X1=R2(RM,1)
11020 IFX1>0THENFORX=0TO128STEP64:PRINT@X+23,CHR$(191)+STRING$(1
6,128)+CHR$(191);:NEXT:ELSEPRINT@192+24,STRING$(16,143);
11025 X1=R2(RM,2)
11030 IFX1>0THENFORX=832TO960STEP64:PRINT@X+23,CHR$(191)+STRING$(
16,128)+CHR$(191);:NEXT:ELSEPRINT@768+24,STRING$(16,188);
11035 X1=R2(RM,3)
11040 IFX1>0PRINT@320+48,STRING$(16,188);:PRINT@640+48,STRING$(1
6,143);:ELSEFORX=384TO576STEP64:PRINT@X+47,CHR$(191);:NEXT
11045 X1=R2(RM,4)
11050 IFX1>0PRINT@320,STRING$(16,188);:PRINT@640,STRING$(16,143)
;:ELSEFORX=384TO576STEP64:PRINT@X+16,CHR$(191);:NEXT
11055 RETURN

```

Subroutines for moving player around screen.

```

15100 IFY5=0THEN15105ELSEM=PEEK(15360+X5+((Y5-1)*64)):IFM=320RM=
128THEN15105ELSERETURN
15105 PRINT@X5+64*Y5, " ";
15110 Y5=Y5-1:IFY5<1THENRM=R2(RM,1):GOTO1000
15120 PRINT@X5+64*Y5,YY%;:RETURN
15200 IFY5=15THEN15205ELSEM=PEEK(15360+X5+((Y5+1)*64)):IFM<>32AN
DM<>128THENRETURN
15205 PRINT@X5+64*Y5, " ";
15210 Y5=Y5+1:IFY5>14THENRM=R2(RM,2):GOTO1000
15220 PRINT@X5+64*Y5,YY%;:RETURN
15300 IFX5>61THEN15305ELSEM=PEEK(15360+X5+1+64*Y5):IFM<>32ANDM<>
128THENRETURN
15305 M=PEEK(15360+X5+2+64*Y5):IFM<>32ANDM<>128THENRETURN
15310 PRINT@X5+64*Y5, " ";
15310 X5=X5+2:IFX5>61THENRM=R2(RM,3):GOTO1000
15320 PRINT@X5+64*Y5,YY%;:RETURN
15400 IFX5<2THEN15405ELSEM=PEEK(15360+X5-1+64*Y5):IFM<>32ANDM<>1
28THENRETURN
15405 M=PEEK(15360+X5-2+64*Y5):IFM<>32ANDM<>128THENRETURN
15410 PRINT@X5+64*Y5, " ";
15410 X5=X5-2:IFX5<2THENRM=R2(RM,4):GOTO1000
15420 PRINT@X5+64*Y5,YY%;:RETURN

```

Normal arrow firing routine.

```

15500 GOSUB15699
15505 X=RND(0)/2:IFRC=1THENX=X-.1
15506 IFRC=2THENX=X+.1
15507 X=X-(EP/1000)
15510 X=X-.2
15511 X=X-(DX/100)
15515 IFX>WTHENRETURN
15520 X=RND(0):IFRC=1THENX=X+.2ELSEX=X+.1
15521 IFRC=2THENX=X-.1
15525 MS=MS-X:RETURN
15599 RETURN

```

Magic arrow firing routine.

```

15600 GOSUB 15699
15601 IFM1(RM)=8THENRETURN
15605 X=RND(0)/2:IFRC=1THENX=X-.1
15606 X=X-(DX/100)
15607 IFRC=2THENX=X+.1
15608 X=X-(EP/1000)
15610 IFX>WTHENRETURN
15620 X=RND(0):IFRC=1THENX=X+.1
15621 IFRC=2THENX=X-.1
15625 MS=MS-X:RETURN

```

15698 RETURN

Calculate monster range, aim, and shoot arrow graphically.

```

15699 IFWX=0THENWX=31:IFWY=0THENWY=8
15700 X6=X5*2-1:Y6=Y5*3-2:X7=WX*2-1:Y7=WY*3-2
15701 IFX6=X7THENSL=0:X8=X7:X9=X6ELSESLP=(Y6-Y7)/(X6-X7):X8=X6:X
9=X7
15702 GOTO 15706
15705 IFX6>X7THENSLP=(Y6-Y7)/(X6-X7):X8=X6:X9=X7:ELSEIFX7>X6THEN
SLP=(Y7-Y6)/(X7-X6):X8=X7:X9=X6:ELSESL=0:X8=X7:X9=X6
15706 Y8=Y6:Y9=Y7
15707 Y=Y8
15708 SL=SL*SGN(Y8-Y9):IFY6<Y7THENSLP=-SLP
15709 IFX6>X7THENSLP=-SLP
15710 FORX=XBTOX9-1STEP64(X9-X8)
15711 IFY>47ORY<0ORX>127ORX<0THENNEXT:GOTO15750
15715 IFPOINT(X,Y)=-1THENX9=X-1:GOTO15750
15720 SET(X,Y):Y=Y+SLP:NEXTX
15750 Y=Y8:FORX=XBTOX9-1STEP64(X9-X8):RESET(X,Y):Y=Y+SLP:NEXTX
15760 RETURN

```

Subroutine for close combat with a monster.

```

16000 IFABS(X5-WX)>1ORABS(Y5-WY)>1THENRETURN
16001 IFM1(RM)=8THENRETURN
16002 IFM1(RM)=7THENRETURN
16003 IFM1(RM)=6THENW=W-.05
16005 X=RND(0):IFRC=0THENX=X-.1
16006 X=X-(DX/100)
16007 IFRC=2THENX=X-.3
16008 X=X-(EP/1000)
16010 IFX>WTHENRETURN
16015 X=RND(0):IFRC=0THENX=X+.1
16016 X=X+(ST/100)
16017 IFRC=2THENX=X+.2
16020 MS=MS-X:RETURN

```

Subroutine for opening a treasure chest.

```

17000 IFABS(TX-X5)>1THENRETURN
17005 IFABS(TY-Y5)>1THENRETURN
17010 PRINT@TX+64*TY, " ";
17011 TX=0:TY=0
17015 PRINT@832+41,T*(T1(RM));:FORX=1TO1000:NEXT
17020 FORX=41TO63:PRINT@832+X, " ";:NEXT
17021 IFT1(RM)=10THENPT=PT+1:GOTO17026ELSEIFT1(RM)=11THENA2=A2+1
0:GOTO17026ELSEIFT1(RM)=12THENA1=A1+10:GOTO17026
17024 TS(T1(RM))=TS(T1(RM))+1
17025 GP=GP+GP(T1(RM))
17026 EP=EP+EP(T1(RM)):T1(RM)=0
17030 RETURN

```

Subroutine to throw a flask of holy water.

```

18000 M=M1(RM):IFM=20RM=50RM=7THENRETURN
18005 GOSUB15699:PRINT@WX+(64*WY), " ";:WA=WX:WB=WY:WX=X5:WY=Y5:G
OSUB16000:WX=WA:WY=WB:PRINT@X5+(64*Y5),YY%;:RETURN

```

Take care of "end of quest" procedures such as saving games on tape and printing out information on the player's fighter.

```

20000 INPUT"Would you like to see the treasures you retrieved";A
%:IFLEFT$(A%,1)="Y"THENFORX=1TO9:PRINTT$(X),"Number retrieved: "
;TS(X):NEXT
20010 FORX=1TO9:TS(X)=0:NEXT
20011 A1=A1+A3:A2=A2+A4

```

continued on next page

continued from previous page

```

20015 INPUT"Would you like to save this game";A$:IFLEFT$(A$,1)<>
"Y"THEN20028
20018 INPUT"To cassette or disk ";A$:IF LEFT$(A$,1)="C"THENINPUT
"PRESS ENTER TO BEGIN SAVE ";A$:GOTO20024
20019 IFLEFT$(A$,1)<>"D"THEN20018
20020 OPEN"O",1,"QUEST/DAT"
20021 FORX=1TO58:PRINT#1,M1(X);M2(X);T1(X):NEXT
20022 CLOSE:GOTO20027
20024 FORX=1TO58:PRINT#-1,M1(X),M2(X),T1(X):NEXT
20027 PRINT"SAVE COMPLETE"
20028 INPUT"Would you like to stop now";A$:IFLEFT$(A$,1)<>"Y"THE
NRETURN
20030 PRINT"Ok. So that you can use this character again:"
20035 PRINT"Name: ";NM$;" Race: ";:IFRC=0THENPRINT"Human":ELSEIF
RC=1THENPRINT"Elf":ELSEPRINT"Dwarf"
20040 PRINT"Wounds: ";W#100;"%"
20045 PRINT"Healing potions: ";PT;" Holy water: ";HW
20050 PRINT"Arrows: ";A1;" Magic Arrows: ";A2
20055 PRINT"Gold: ";GP;" Experience: ";EP
20060 PRINT"Strength: ";ST;" Dexterity: ";DX
20065 INPUT"Would you like to try again as a *NEW* character";A$
:IFLEFT$(A$,1)="Y"THENRUN
20099 PRINT:PRINT"Come Questing again sometime!!!":END
    
```

Subroutine to create new fighter characters.

```

21000 PRINT"Ok, I'll make you one.":FORX=1TO1000:NEXT
21005 GP=RND(20)+5:ST=RND(17)+3:DX=RND(17)+3:RC=RND(3)-1:A1=3:A2
=RND(10):PT=RND(3)+1:HW=RND(5):EP=0:W=1
21010 PRINT"Strength: ";ST;" Dexterity: ";DX
21015 PRINT"Gold: ";GP;" Healing potions: ";PT
21020 PRINT"Holy water: ";HW;" Race: ";:IFRC=1THENPRINT"Elf"ELS
EIFRC=2PRINT"Dwarf"ELSEPRINT"Human"
21025 PRINT"Arrows: ";A1;" Magic arrows: ";A2
21027 IFTRSBOMODEL=3THENINPUT"Is this character female";A$:IFLEF
T$(A$,1)="Y"THENYY$=CHR$(254)
21028 IFTRSBOMODEL=3THENPOKE16409,0
21030 INPUT"What will you name this character";NM$:PRINT"Have a
fun Quest, ";NM$;"!!!":FORX=1TO1000:NEXT:CLS
21035 IFTRSBOMODEL=3THENPOKE16409,1
21040 RETURN
    
```

Error-handling routine (for arrow-shooting FC errors).

```

29999 END
30000 IFERL=15750THENRESUME15760ELSEIFERL=4096THENRESUMENEXTELSE
PRINTERL,ERR/2+1:FORX=1TO1000:NEXT:RESUMENEXT
    
```

ATARI VERSION

NOTE: User must hit [SYSTEM RESET] prior to every "RUN" of "Quest 1" to reinitialize the character set.

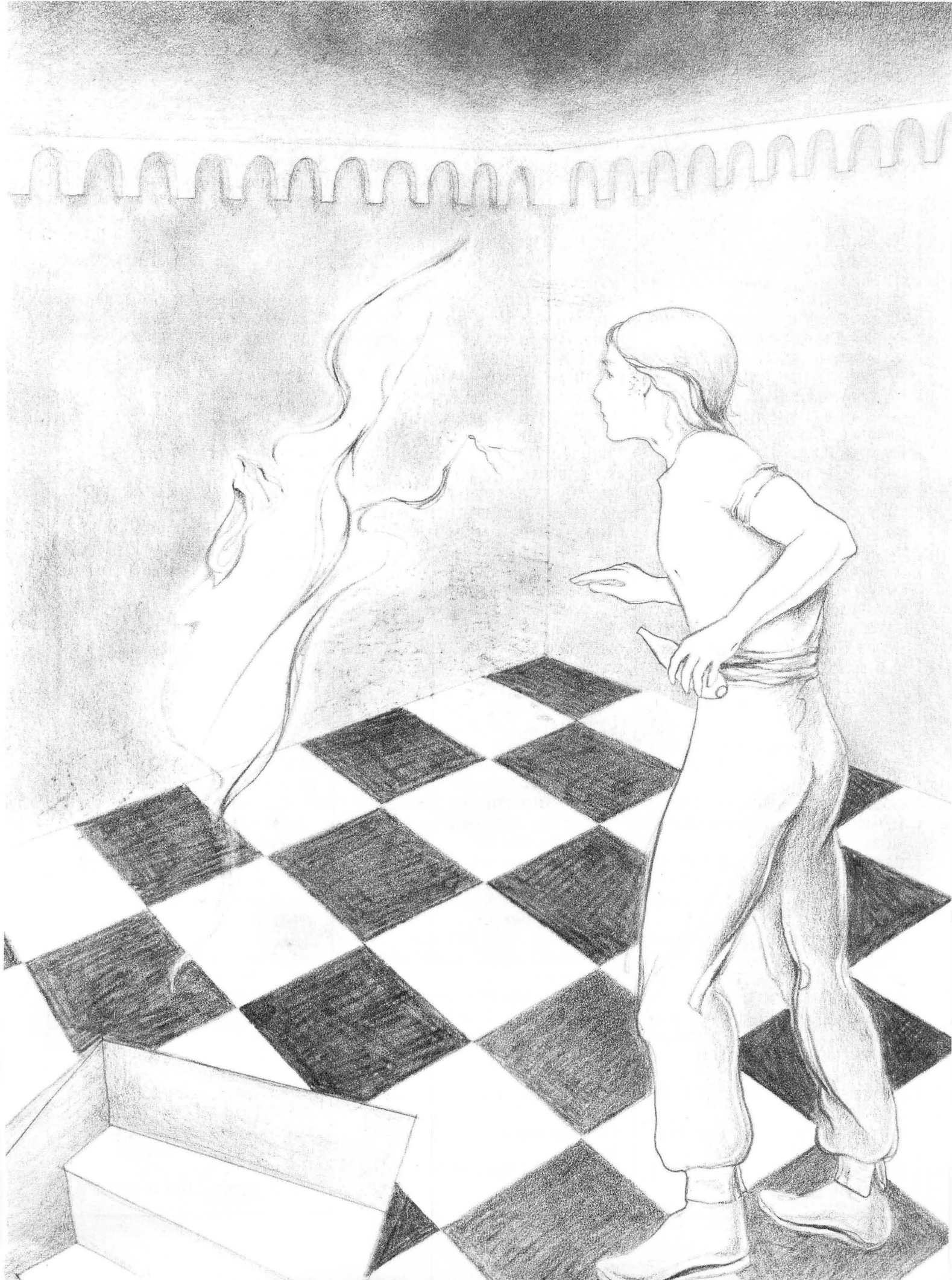
Lines 1-9: Initialize character modification and print title page.
 Lines 100-112: Data for monsters and treasures.
 Lines 115-590: Data for the rooms.
 Lines 600-620: Initialize variables.
 Lines 800-846: Ask if you want to use an old character.
 850-884: Ask if you want to load in an old game.
 Lines 900-980: Marketplace and bargaining routine.
 Lines 990-999: Enter dungeon; check for too many arrows.
 Lines 1000-1050: Upon entering a new

room, draw it with its monsters and treasures. If this is room one, give option to leave.
 Lines 1055-1088: Print player status; check for wandering monsters.
 Lines 1090-1145: Accept a command from the keyboard and call appropriate subroutine.
 Lines 1200-1250: If there is a monster in the room, move him and let him attack.
 Lines 5000-5025: End-routine for the "Great Dungeon in the Sky" ending.
 Lines 10000-10045: Subroutine to draw a passage/intersection.
 Lines 11000-11055: Subroutine to draw a chamber/room.
 Lines 15100-15420: Subroutine for moving player around screen.
 Lines 15500-15599: Normal arrow firing

routine.
 Lines 15600-15698: Magic arrow firing routine.
 Lines 15699-15760: Calculate monster range, aim, and shoot arrow graphically.
 Lines 16000-16020: Subroutine for close combat with a monster.
 Lines 17000-17030: Subroutine for opening a treasure chest.
 Lines 18000-18005: Subroutine to throw a flask of holy water.
 Lines 20000-20099: Take care of "end of quest" procedures such as saving games on tape or disk, and printing out information on the player's fighter.
 Lines 21000-21040: Subroutine to create new fighter characters.
 Lines 30000-30050: Redefines "&" to be a red block and "@" to be a man character.

<pre> 1 GOSUB 30000: ? CHR\$(125):POKE 752,1:P OSITION 15,6: ? " QUEST 1 " 2 ? : ? "QUEST 1 was written by Brian R eynolds":OPEN #3,4,0,"K" 3 ? : ? " (ATARI translation by Alan J. Zett)" 7 GOSUB 600:FOR X=1 TO 276:T\$(X)=" ":N EXT X:FOR X=1 TO 88:MN\$(X)=" ":NEXT X: YY\$="" 8 FOR X=0 TO 9:TS(X)=0:NEXT X 9 FOR X=1 TO 49:K\$(X,X)=CHR\$(INT(RND(0)*32)):NEXT X:K\$(49)=" " 100 DATA Worthless odds and ends,0,0,A bag of Copper Coins,1,3,A small Brass Statuette,2,5 102 DATA A bag of various Coins,3,7,A purse of Gold Coins,5,12,3 Gold Nugget s,8,17 104 DATA 4 small Turquoises,7,15,A lar ge Ruby,15,30 </pre>	<pre> 106 DATA A *HUGE* Sapphire,150,150,A H ealing Potion,10,0 108 DATA 10 Magic Arrows,15,0,10 norma l Arrows,10,0 110 DATA SKELETON,S,2,ORC,0,3,ZOMBIE,Z ,4,GHOUL,g,6 112 DATA HUGE SPIDER,H,7,MUMMY,M,8,GIA NT,G,9,WRAITH,W,9.9 115 DATA 1,12,3,2,18,0,0,0 120 DATA 2,0,0,0,1,4,2,8 125 DATA 1,1,0,4,19,0,0,1 130 DATA 1,0,0,5,3,3,1,1 135 DATA 2,6,38,0,4,1,3,6 140 DATA 1,8,5,9,7,0,0,0 145 DATA 1,0,0,6,0,0,0,1 150 DATA 2,0,6,0,11,2,11,2 155 DATA 2,0,0,10,6,2,3,1 160 DATA 2,0,0,0,9,5,1,4 165 DATA 1,0,0,8,12,0,0,1 170 DATA 2,0,1,11,13,2,5,3 </pre>	<pre> 175 DATA 1,0,0,12,14,0,0,1 180 DATA 2,15,26,13,17,5,1,1 185 DATA 2,0,14,0,0,0,0,1 190 DATA 2,0,17,0,0,1,2,5 195 DATA 1,16,20,14,0,4,1,1 200 DATA 2,0,19,1,26,2,2,7 205 DATA 2,18,30,3,27,3,2,2 210 DATA 1,17,21,0,0,0,0,1 215 DATA 1,20,22,0,0,6,2,9 220 DATA 1,21,23,0,0,2,3,12 225 DATA 1,22,24,0,0,4,2,10 240 DATA 1,23,25,34,0,0,0,11 250 DATA 2,24,0,0,0,7,3,9 260 DATA 2,14,0,18,0,3,2,1 270 DATA 2,0,28,19,0,4,1,2 280 DATA 1,27,29,31,0,0,0,1 290 DATA 2,28,0,0,0,5,1,10 300 DATA 2,19,0,0,0,1,2,3 310 DATA 1,0,32,0,28,0,0,4 </pre>
--	--	--

continued on page 32



continued from page 30

```
320 DATA 1,31,33,43,0,0,0,1
330 DATA 2,32,35,0,0,5,1,8
340 DATA 1,0,0,35,24,0,0,12
350 DATA 1,33,36,45,34,0,0,5
360 DATA 1,35,0,37,0,7,1,10
370 DATA 2,0,0,0,36,8,3,9
380 DATA 1,5,49,0,39,0,0,1
390 DATA 1,0,40,38,0,0,0,6
400 DATA 1,39,0,0,41,2,3,2
410 DATA 1,42,46,40,43,4,1,7
420 DATA 2,0,41,0,0,7,3,8
430 DATA 2,0,44,41,32,6,1,11
440 DATA 1,43,45,0,0,0,0,5
450 DATA 1,44,0,47,35,0,0,1
460 DATA 2,41,47,48,0,5,1,7
470 DATA 1,46,0,50,45,0,0,3
480 DATA 1,0,0,49,46,0,0,1
490 DATA 2,38,51,52,48,6,1,6
500 DATA 1,0,0,51,47,2,5,10
510 DATA 1,49,0,53,50,4,3,5
520 DATA 2,0,0,0,49,6,1,6
530 DATA 2,0,54,0,51,5,1,8
540 DATA 1,53,0,0,55,0,0,1
550 DATA 2,0,0,54,56,2,3,2
560 DATA 1,0,0,55,57,6,1,8
570 DATA 1,0,0,56,58,7,3,11
580 DATA 2,0,0,57,0,8,4,9
590 RESTORE :GOTO 603
600 DIM MN$(89),M$(8),MS(8),R1(58),R2(
58,4),M1(58),M2(58),T1(58),T$(277),EP(
12),GP(12),TS(9),YY$(1)
602 DIM X$(23),NM$(20),A$(5),K$(49),X1
$(2),I$(2):RETURN
603 FOR X=1 TO 12:READ X$,X1,X2:T$(X$
23)-22=X$:EP(X)=X1:GP(X)=X2:NEXT X:T$
(277)="#"
605 FOR X=1 TO 8:READ X$,X1$,X2:MN$(X
$11)-10=X$:M$(X)=X1$:MS(X)=X2:NEXT X:
MN$(89)="#"
610 FOR X=1 TO 58:READ X1:R1(X)=X1:FOR
Y=1 TO 4:READ X1:R2(X,Y)=X1:NEXT Y
615 READ X1:M1(X)=X1:READ X1:M2(X)=X1:
READ X1:T1(X)=X1:NEXT X
620 RM=1:A1=1000:A2=1000:W=1:P2=2:POKE
752,0
800 IF B1=1 THEN GOSUB 20000
805 IF B1=1 THEN B1=0:GOTO 900
810 POSITION 2,12:? "Want to use an ol
d character ";;INPUT A$?: IF A$(1,1)<
>"Y" THEN GOSUB 21000:GOTO 900
812 ? "NAME: ";;INPUT NM$
815 ? "STRENGTH: ";;INPUT ST:IF ST>20
OR ST<3 THEN 815
820 ? "DEXTERITY: ";;INPUT DX:IF DX>20
OR DX<3 THEN 820
825 ? "WOUNDS: ";;INPUT W:W=W/100:IF W
<0.1 OR W>1 THEN 825
```

```
830 ? "EXPERIENCE: ";;INPUT EP
832 ? "GOLD: ";;INPUT GP
835 ? "IS (S)HE AN ELF ";;INPUT A$:IF
A$(1,1)="Y" THEN RC=1
836 IF RC=0 THEN ? "IS (S)HE A DWARF "
;;INPUT A$:IF A$(1,1)="N" THEN RC=2
840 ? "MAGIC ARROWS: ";;INPUT A2:? "NO
RMAL ARROWS: ";;INPUT A1
845 ? "HEALING POTIONS: ";;INPUT PT
846 ? "HOLY WATER: ";;INPUT HW
850 ? "Want to load in an old game ";;
INPUT A$:IF A$(1,1)<>"Y" THEN 900
860 ? "FROM CASSETTE OR DISK ";;INPUT
A$:IF A$(1,1)="C" THEN ? "HIT <RETURN>
WHEN READY ":GOTO 880
862 IF A$(1,1)<>"D" THEN 860
870 OPEN #2,4,0,"D:QUEST.DAT"
872 FOR X=1 TO 58:INPUT #2;X1:M1(X)=X1
:INPUT #2;X1:M2(X)=X1:INPUT #2;X1:T1(X
)=X1
874 GOTO 886
880 OPEN #2,4,0,"C:QUEST.DAT"
882 FOR X=1 TO 58:INPUT #2;X1:M1(X)=X1
:INPUT #2;X1:M2(X)=X1:INPUT #2;X1:T1(X
)=X1
884 NEXT X:CLOSE #2
900 ? CHR$(125);"GOLD: ";GP
901 ? :? "You're at a market. Prices h
ere are:":?
903 ? "[1] MAGIC ARROW ..... 2
GOLD [2] FOUR NORMAL ARROWS ..... 1
GOLD"
905 ? "[3] HEALING POTION ..... 15
GOLD [4] HOLY WATER ..... 3
GOLD"
910 ? :? "OK, ";NM$;," what do you nee
d ";;INPUT A$:IT=VAL(A$)
911 IF IT>4 OR IT<0 THEN ? CHR$(253);"
I DON'T SELL THAT!":GOTO 910
912 IF IT=0 THEN 990
913 IF IT=1 THEN P1=2
914 IF IT=2 THEN P1=1
915 IF IT=3 THEN P1=15
916 IF IT=4 THEN P1=3
917 ? :? "At ";P1;" GOLD apiece,":? "h
ow many will you buy ";;INPUT A$:NM=VA
L(A$):PRINT
918 IF NM<0 THEN PRINT CHR$(253);"VERY
FUNNY!":PRINT "I DON'T BUY THINGS, I
SELL THEM!":GOTO 916
920 P1=P1#NM
921 OP=P1
925 ? :? "That comes to ";P1;" GOLD, "
;NM$
930 ? "How much will you give me ";;IN
PUT A?:
935 IF A<OP/10 THEN ? "FORGET IT!!!":G
OTO 901
```

```
940 IF A<OP/2 THEN ? "NOT INTERESTED."
:GOTO 901
941 IF A>P1 THEN ? "YOU GOT A DEAL!":
GOTO 950
942 Y=A/P1:X=RND(0):IF X>Y THEN ? "Not
interested.":P1=INT((OP+P1)/2):GOTO 9
30
945 P1=INT((P1*2+A)/3):IF P1<=A THEN 9
41
947 ? "How about ";P1;," ";NM$;"?":GOT
O 930
950 IF GP<P1 THEN ? :? "WHAT!! YA CAN'
T PAY YER DEBTS!":? "YOU'LL BE THROWN
INTO PRISON FOR THIS!";CHR$(253):END
955 GP=GP-P1:? "You now have ";GP;" GO
LD, ";NM$
957 IF IT=4 THEN HW=HW+NM
960 IF IT=1 THEN A2=A2+NM
965 IF IT=2 THEN A1=A1+NM*4
970 IF IT=3 THEN PT=PT+NM
980 GOTO 901
990 ? "OK, ";NM$;," PRESS <RETURN> TO"
:? "ENTER THE DUNGEON!":
991 EL=0
992 IF EP>100 THEN EP=EP-100:EL=EL+100
:FOR X=1 TO 58:M2(X)=M2(X)*1.1:NEXT X:
GOTO 992
993 EP=EP+EL
994 IF EL>500 THEN FOR U=EL TO 500 STE
P -100:FOR X=1 TO 58:M2(X)=M2(X)/1.1:N
EXT X:NEXT U
995 INPUT A$:? CHR$(125):POKE 752,1
996 A3=0:A4=0
997 IF A2>ST*2 THEN A4=A2-ST*2:A2=ST*2
:? "MORE THAN ";ST*2;" MAGIC ARROWS WO
ULD":? "WEIGH YOU DOWN":?
998 IF A1>ST*2 THEN A3=A1-ST*2:A1=ST*2
:? "MORE THAN ";ST*2;" ARROWS WOULD":?
"WEIGHT YOU DOWN":?
999 FOR X=1 TO 500:NEXT X
1000 POKE 752,1:? CHR$(125):IF B1=0 TH
EN B1=1:GOTO 1005
1001 IF RM<>1 THEN 1005
1002 PRINT "Do you wish to leave the d
ungeon ";;INPUT A$
1003 IF A$(1,1)="Y" THEN 800
1004 ? CHR$(125)
1005 ON R1(RM) GOSUB 10000,11000
1010 IF T1(RM)>0 THEN TX=INT(RND(0)*9)
+16:TY=INT(RND(0)*6)+9:POSITION TX,TY:
PRINT "#";
1015 X5=20:Y5=11
1020 IF I$="W" THEN Y5=22
1022 IF I$="X" THEN Y5=2
1024 IF I$="D" THEN X5=2
1026 IF I$="A" THEN X5=38
1028 POSITION X5,Y5:? YY$;
1030 IF M2(RM)>=1 THEN WX=INT(RND(0)*9
```

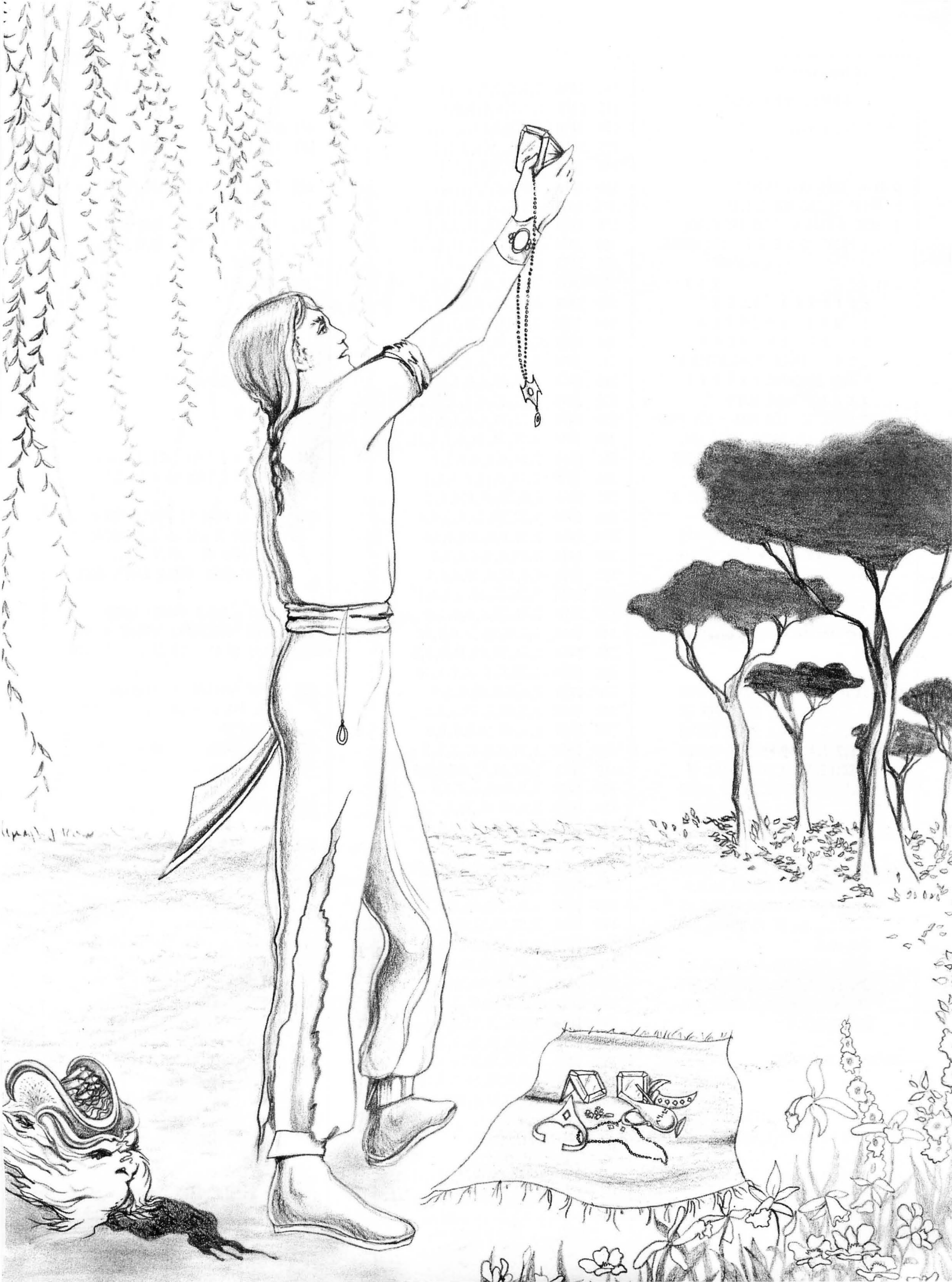

continued from previous page

```
15403 LOCATE X5-2,Y5,M:POSITION X5-2,Y
5:? CHR$(M):IF M<>32 THEN RETURN
15405 POSITION X5,Y5:? " ";
15410 X5=X5-2:IF X5<3 THEN RM=R2(RM,4)
:GOTO 1000
15420 GOTO 15120
15500 GOSUB 15699
15505 X=RND(0)/2:IF RC=1 THEN X=X-0.1
15506 IF RC=2 THEN X=X+0.1
15507 X=X-(EP/1000)
15510 X=X-0.2
15511 X=X-(DX/100)
15515 IF X>W THEN RETURN
15520 X=RND(0):IF RC=1 THEN X=X+0.2
15522 IF RC<>1 THEN X=X+0.1
15523 IF RC=2 THEN X=X+0.1
15525 MS=MS-X:RETURN
15599 RETURN
15600 GOSUB 15699
15601 IF M1(RM)=8 THEN RETURN
15605 X=RND(0)/2:IF RC=1 THEN X=X-0.1
15606 X=X-(DX/100)
15607 IF RC=2 THEN X=X+0.1
15608 X=X-(EP/1000)
15610 IF X>W THEN RETURN
15620 X=RND(0):IF RC=1 THEN X=X+0.1
15621 IF RC=2 THEN X=X-0.1
15625 MS=MS-X:RETURN
15698 RETURN
15699 IF WX=0 THEN WX=31:IF WY=0 THEN
WY=8
15700 X6=X5:Y6=Y5-1:X7=WX:Y7=WY
15701 IF X6=X7 THEN SL=0:X8=X7:X9=X6
15702 IF X6<>X7 THEN SL=(Y6-Y7)/(X6-X7
):X8=X6:X9=X7
15703 GOTO 15708
15705 IF X6>X7 THEN SL=(Y6-Y7)/(X6-X7)
:X8=X6:X9=X7
15706 IF X7>X6 THEN SL=(Y7-Y6)/(X7-X6)
:X8=X7:X9=X6
15707 IF X7=X6 THEN SL=0:X8=X7:X9=X6
15708 Y8=Y6:Y9=Y7
15709 Y=Y8
15710 SL=SL*SGN(Y8-Y9):IF Y6<Y7 THEN S
L=-SL
15711 IF X6>X7 THEN SL=-SL
15712 FOR X=X8 TO X9 STEP SGN(X9-X8+0.
01):FOR XX=1 TO 5:NEXT XX
15713 IF Y>22 OR Y<1 OR X>37 OR X<3 TH
EN NEXT X:GOTO 15750
15715 LOCATE X,Y,A:IF A=38 THEN X9=X-1
:GOTO 15750
15720 POSITION X,Y:PRINT "+":Y=Y+SL:N
EXT X
15750 Y=Y8:FOR X=X8 TO X9 STEP SGN(X9-
X8+0.01):POSITION X,Y:PRINT " ":Y=Y+5
L:NEXT X
```

```
15760 RETURN
16000 IF ABS(X5-WX)>1 OR ABS(Y5-WY)>1
THEN RETURN
16001 IF M1(RM)=8 THEN RETURN
16002 IF M1(RM)=7 THEN RETURN
16003 IF M1(RM)=6 THEN W=W-0.05
16005 X=RND(0):IF RC=0 THEN X=X-0.1
16006 X=X-(DX/100)
16007 IF RC=2 THEN X=X-0.3
16008 X=X-(EP/1000)
16010 IF X>W THEN RETURN
16015 X=RND(0):IF RC=0 THEN X=X+0.1
16016 X=X+(ST/100)
16017 IF RC=2 THEN X=X+0.2
16020 MS=MS-X:RETURN
17000 IF ABS(TX-X5)>1 THEN RETURN
17005 IF ABS(TY-Y5)>1 THEN RETURN
17010 POSITION TX,TY:? " ";
17011 TX=0:TY=0
17015 POSITION 12,23:? T*((T1(RM)*23)-
22,T1(RM)*23):FOR X=1 TO 120:NEXT X
17020 FOR X=12 TO 34:POSITION X,23:? "
":NEXT X
17021 IF T1(RM)=10 THEN PT=PT+1:GOTO 1
7026
17022 IF T1(RM)=11 THEN A2=A2+10:GOTO
17026
17023 IF T1(RM)=12 THEN A1=A1+10:GOTO
17026
17024 T5(T1(RM))=T5(T1(RM))+1
17025 GP=GP+GP(T1(RM))
17026 EP=EP+EP(T1(RM)):T1(RM)=0
17030 RETURN
18000 M=M1(RM):IF M=2 OR M=5 OR M=7 TH
EN RETURN
18005 GOSUB 15699:POSITION WX,WY:? " "
:WA=WX:WB=WY:WX=X5:WY=Y5:GOSUB 16000:
WX=WA:WY=WB:POSITION X5,Y5:? YY$:RETU
RN
20000 PRINT "WOULD YOU LIKE TO SEE THE
TREASURES YOU RETRIEVED FROM THE DU
NGEON":INPUT A$
20005 IF A$(1,1)="Y" THEN FOR X=1 TO 9
:? STR$(X);". ";T*((X*23)-22,X*23):CHR
$(127):TS(X):NEXT X
20010 FOR X=1 TO 9:TS(X)=0:NEXT X
20011 A1=A1+A3:A2=A2+A4
20015 ? :? "WOULD YOU LIKE TO SAVE THI
S GAME ":INPUT A$
20017 IF A$(1,1)<>"Y" THEN 20028
20018 ? "CASSETTE OR DISK ":INPUT A$:
IF A$(1,1)="C" THEN ? "HIT <RETURN> WH
EN READY ":GOTO 20024
20019 IF A$(1,1)<>"D" THEN 20018
20020 OPEN #2,B,0,"D:QUEST.DAT"
20021 FOR X=1 TO 58:PRINT #2;M1(X):PRI
NT #2;M2(X):PRINT #2;T1(X)
20022 GOTO 20027
```

```
20024 OPEN #2,B,0,"C:QUEST.DAT"
20025 FOR X=1 TO 58:PRINT #2;M1(X):PRI
NT #2;M2(X):PRINT #2;T1(X)
20027 NEXT X:CLOSE #2:? "SAVE COMPLETE
."
20028 ? "Would you like to stop now ";
:INPUT A$:IF A$(1,1)<>"Y" THEN RETURN
20030 ? "OK. So that you can use this
character again at a later time:"
20035 ? "NAME: ";NM$;" RACE: ";:IF RC=
0 THEN ? "HUMAN"
20036 IF RC=1 THEN ? "ELF"
20037 IF RC=2 THEN ? "DWARF"
20045 ? "HEALING POTIONS: ";PT
20046 ? "HOLY WATER: ";HW
20050 ? "ARROWS: ";A1;" MAGIC ARROWS
":A2
20055 ? "GOLD: ";GP;" EXPERIENCE: ";
EP
20060 ? "STRENGTH: ";ST;" DEXTERITY:
":DX
20065 ? :? "Would you like to try agai
n":? "as a *NEW* character ":INPUT A$
:IF A$(1,1)="Y" THEN CLR :GOTO 7
20099 GRAPHICS 0:? "COME QUESTIONING AGAI
N SOMETIME!!":END
21000 ? "OK, I'll make you one.":FOR X
=1 TO 100:NEXT X:GP=INT(RND(0)*20)+6:5
T=INT(RND(0)*17)+4:DX=INT(RND(0)*17)+4
21005 RC=INT(RND(0)*3):A1=3:A2=INT(RND
(0)*10)+1:PT=INT(RND(0)*3)+2:HW=INT(RN
D(0)*5)+1:EP=0:W=1
21010 ? "STRENGTH: ";ST;" DEXTERITY:
":DX
21015 ? "GOLD: ";GP;" HEALING POTION
S: ";PT
21020 ? "HOLY WATER: ";HW;" RACE: ";
:IF RC=0 THEN ? "HUMAN"
21021 IF RC=1 THEN ? "ELF"
21022 IF RC=2 THEN ? "DWARF"
21025 ? "ARROWS: ";A1;" MAGIC ARROWS
":A2
21030 ? :? "What will you name this ch
aracter":INPUT NM$:?:? "HAVE A FUN QU
EST, ";NM$;"!!!!"
21040 FOR X=1 TO 200:NEXT X:?:CHR$(125
):RETURN
30000 POKE 106,PEEK(106)-5:GRAPHICS 0:
? "INITIALIZING . . .":SETCOLOR 2,7,0
30010 START=(PEEK(106)+1)*256
30020 FOR X=0 TO 1023:POKE START+X,PEE
K(57344+X):NEXT X:RESTORE 30050
30030 POKE 756,START/256
30040 FOR X=0 TO 7:POKE X+START+48,85:
NEXT X:FOR X=0 TO 7:READ X1:POKE X+STA
RT+256,X1:NEXT X:RETURN
30050 DATA 152,216,255,27,25,60,102,23
1
```

continued on page 36



APPLE VERSION

Print title page.

```

0 D$ = CHR$ (4): PRINT
1 PRINT D$;"NO MON C,I,D"
2 HOME : VTAB 6: HTAB 14: FLASH
  : PRINT "Q U E S T 1": NORMAL

3 A$ = "          * * *
  * * * * * Q U E S T
  1 W A S W R I T T E N
  B Y B R I A N R E Y N O
  L D S -- APPLE TRANSLATION B
  Y R I C H B O U C H A R D * * * * *
  * * * * (HAVE FUN!) "

4 FOR X = 1 TO LEN (A$) - 13: VTAB
  9: HTAB 14: PRINT MID$ (A$,
  X,13); FOR Y = 1 TO 75: NEXT
  Y: NEXT X

5 DEF FN A(X) = INT ( RND (1) *
  X + 1)

6 YY$ = "Q":K$ = "!%$%&'*(:/<+>#
  ":K$ = K$ + K$ + K$ + "****+
  +++ "
    
```

Data for monsters and treasures.

```

100 DATA WORTHLESS ODDS & ENDS
  ,0,0,A BAG FULL OF COPPER CO
  INS,1,3,A SMALL BRASS STATUE
  TTE,2,5,A BAG FULL OF VARIQU
  S COINS,3,7,A PURSE FULL OF
  GOLD COINS,5,12,3 GOLD NUGGE
  TS ,8,17,4 SMALL TURQUOI
  SES,7,15,A LARGE RUBY ,1
  5,30

105 DATA A *HUGE* SAPPHIRE,150,
  150,A HEALING POTION,10,0,A
  QUIVER OF 10 MAGIC ARROWS,15
  ,0,A QUIVER OF 10 NQRMAL ARR
  QWS,10,0

110 DATA SKELETON,S,2,ORC,0,3,Z
  OMBIE,Z,4,GHOUL,G,6,HUGE SPI
  DER,H,7,MUMMY,M,8,GIANT,G,9,
  WRAITH,W,9.9
    
```

Data for the rooms.

```

115 DATA 1,12,3,2,18,0,0,0
120 DATA 2,0,0,0,1,4,2,8
125 DATA 1,1,0,4,19,0,0,1
130 DATA 1,0,0,5,3,3,1,1
135 DATA 2,6,38,0,4,1,3,6
    
```

```

140 DATA 1,8,5,9,7,0,0,0
145 DATA 1,0,0,6,0,0,0,1
150 DATA 2,0,6,0,11,2,11,2
155 DATA 2,0,0,10,6,2,3,1
160 DATA 2,0,0,0,9,5,1,4
165 DATA 1,0,0,8,12,0,0,1
170 DATA 2,0,1,11,13,2,5,3
175 DATA 1,0,0,12,14,0,0,1
180 DATA 2,15,26,13,17,5,1,1
185 DATA 2,0,14,0,0,0,0,1
190 DATA 2,0,17,0,0,1,2,5
195 DATA 1,16,20,14,0,4,1,1
200 DATA 2,0,19,1,26,2,2,7
205 DATA 2,18,30,3,27,3,2,2
210 DATA 1,17,21,0,0,0,0,1
215 DATA 1,20,22,0,0,6,2,9
220 DATA 1,21,23,0,0,2,3,12
225 DATA 1,22,24,0,0,4,2,10
240 DATA 1,23,25,34,0,0,0,11
250 DATA 2,24,0,0,0,7,3,9
260 DATA 2,14,0,18,0,3,2,1
270 DATA 2,0,28,19,0,4,1,2
280 DATA 1,27,29,31,0,0,0,1
290 DATA 2,28,0,0,0,5,1,10
300 DATA 2,19,0,0,0,1,2,3
310 DATA 1,0,32,0,28,0,0,4
320 DATA 1,31,33,43,0,0,0,1
330 DATA 2,32,35,0,0,5,1,8
340 DATA 1,0,0,35,24,0,0,12
350 DATA 1,33,36,45,34,0,0,5
360 DATA 1,35,0,37,0,7,1,10
370 DATA 2,0,0,0,36,8,3,9
380 DATA 1,5,49,0,39,0,0,1
390 DATA 1,0,40,38,0,0,0,6
400 DATA 1,39,0,0,41,2,3,2
410 DATA 1,42,46,40,43,4,1,7
420 DATA 2,0,41,0,0,7,3,8
430 DATA 2,0,44,41,32,6,1,11
440 DATA 1,43,45,0,0,0,0,5
450 DATA 1,44,0,47,35,0,0,1
460 DATA 2,41,47,48,0,5,1,7
470 DATA 1,46,0,50,45,0,0,3
480 DATA 1,0,0,49,46,0,0,1
490 DATA 2,38,51,52,48,6,1,6
500 DATA 1,0,0,51,47,2,5,10
510 DATA 1,49,0,53,50,4,3,5
520 DATA 2,0,0,0,49,6,1,6
530 DATA 2,0,54,0,51,5,1,8
540 DATA 1,53,0,0,55,0,0,1
550 DATA 2,0,0,54,56,2,3,2
560 DATA 1,0,0,55,57,6,1,8
570 DATA 1,0,0,56,58,7,3,11
580 DATA 2,0,0,57,0,8,4,9
    
```

Initialize variables.

```

600 DIM MN$(8),M$(8),MS(8),R1(58
  ),R2(58,4),M1(58),M2(58),T1(
    
```

```

  58),T$(12),EP(12),GP(12),TS(
  12)
601 BELL$ = CHR$ (7)
603 FOR X = 1 TO 12: READ T$(X),
  EP(X),GP(X): NEXT X
605 FOR X = 1 TO 8: READ MN$(X),
  M$(X),MS(X): NEXT X
610 FOR X = 1 TO 58: READ R1(X):
  FOR Y = 1 TO 4: READ R2(X,Y
  ): NEXT Y
615 READ M1(X),M2(X),T1(X): NEXT
  X
620 RM = 1
    
```

Use an old character?

```

800 IF B1 = 1 THEN GOSUB 20000
805 IF B1 = 1 THEN B1 = 0: GOTO
  900
810 HTAB 1: VTAB 13: INPUT "DO Y
  OU WANT TO USE AN OLD CHARAC
  TER?";A$: IF LEFT$(A$,1) <
  > "Y" THEN GOSUB 21000: GOTO
  900
812 PRINT : INPUT "NAME: ";NM$
815 INPUT "STRENGTH: ";A$:ST = VAL
  (A$): IF ST > 20 OR ST < 3 THEN
  815
820 INPUT "DEXTERITY: ";A$:DX =
  VAL (A$): IF DX > 20 OR DX <
  3 THEN 820
825 INPUT "WOUNDS: ";A$:W = VAL
  (A$) / 100: IF W < .1 OR W >
  1 THEN 825
830 INPUT "EXPERIENCE: ";A$:EP =
  VAL (A$)
832 INPUT "GOLD: ";A$:GP = VAL
  (A$)
835 INPUT "IS (S)HE AN ELF? ";A$
  : IF LEFT$(A$,1) = "Y" THEN
  RC = 1
836 IF RC = 0 THEN INPUT "IS (S
  )HE A DWARF? ";A$: IF LEFT$(
  A$,1) = "N" THEN RC = 2
840 INPUT "MAGIC ARROWS: ";A$:A2
  = VAL (A$): INPUT "NORMAL
  ARROWS: ";A$:A1 = VAL (A$)
845 INPUT "HEALING POTIONS: ";A$
  :PT = VAL (A$)
846 INPUT "HOLY WATER: ";A$:HW =
  VAL (A$)
    
```

Load in an old game?


```

850 PRINT : INPUT "DO YOU WANT T
D LOAD IN AN OLD GAME? ";A$:
IF LEFT$(A$,1) < > "Y" THEN
900
852 INPUT "DISK OR CASSETTE ?";A
$
855 IF LEFT$(A$,1) = "C" THEN
INPUT "HIT <RETURN> TO BEGI
N READ ";A$: PRINT : PRINT "
(WAIT FOR SIX BEEPS.)": RECALL
M1: RECALL M2: RECALL T1: GOTO
890
857 INPUT "ENTER FILESPEC >";A$
860 PRINT D$;"OPEN ";A$
865 PRINT D$;"READ ";A$
870 FOR X = 1 TO 58: INPUT M1(X)
: INPUT M2(X): INPUT T1(X): NEXT
X
875 PRINT D$;"CLOSE ";A$
890 PRINT "LOAD COMPLETE."

```

Marketplace and bargaining routine.

```

900 HOME : PRINT "GOLD: ";GP
901 PRINT : PRINT "YOU ARE AT TH
E MARKET. PRICES HERE ARE:"
903 PRINT " 1. MAGIC ARROW. .
. . . .2 GOLD"
904 PRINT " 2. FOUR NORMAL ARR
OWS . . .1 GOLD"
905 PRINT " 3. HEALING POTION
. . . .15 GOLD"
906 PRINT " 4. HOLY WATER . .
. . . .3 GOLD"
910 PRINT : PRINT "OK, ";NM$;,"
WHAT ITEM WOULD": INPUT "YOU
LIKE? (ENTER NUMBER) ";A$:
IT = VAL (A$): IF IT > 4 OR
IT < 0 THEN PRINT BELL$;"I
DON'T SELL THAT!": GOTO 910
911 IF IT = 0 THEN 990
912 IF IT = 1 THEN P1 = 2
913 IF IT = 2 THEN P1 = 1
914 IF IT = 3 THEN P1 = 15
915 IF IT = 4 THEN P1 = 3
916 PRINT "AT ";P1;" GOLD APIECE
, HOW MANY WILL": INPUT "YOU
BUY? ";A$:NM = VAL (A$)
917 IF NM < 0 THEN PRINT BELL$;
"VERY FUNNY!!": PRINT "I DO
NOT BUY THINGS, I SELL THEM!
": GOTO 916
920 P1 = P1 * NM
921 OP = P1

```

```

925 PRINT "THE PRICE NOW COMES T
O ";P1;" GOLD."
930 PRINT "HOW MUCH WILL YOU GIV
E ME, ";NM$;: INPUT "? ";A$:
A = VAL (A$)
935 IF A < OP / 10 THEN PRINT "
FORGET IT!!!": GOTO 901
940 IF A < OP / 2 THEN PRINT "N
OT INTERESTED.": GOTO 901
941 IF A > = P1 THEN PRINT "YO
U GOT A DEAL!!!": GOTO 950
942 Y = A / P1:X = RND (1): IF X
> Y THEN PRINT "NOT INTERE
STED":P1 = INT ((OP + P1) /
2): GOTO 930
945 P1 = INT ((P1 * 2 + A) / 3):
IF (P1 > = A) THEN 941
947 PRINT "HOW ABOUT ";P1;," ";N
M$;"?": GOTO 930
950 IF GP < P1 THEN PRINT BELL$
;"WHAT!! CAN'T PAY YER DEBT
S, ";NM$;"?","YOU'LL BE THRO
WN IN PRISON FOR THIS!!!";BE
LL$: END
955 GP = GP - P1: PRINT "YOU NOW
HAVE ";GP;" GOLD, ";NM$;."
960 IF IT = 1 THEN A2 = A2 + NM
965 IF IT = 2 THEN A1 = A1 + NM *
4
970 IF IT = 3 THEN PT = PT + NM
975 IF IT = 4 THEN HW = HW + NM
980 GOTO 901

```

Enter dungeon; check for too many arrows.

```

990 PRINT "OK, ";NM$;," PRESS <R
ETURN> TO GO","INTO THE DUNG
EON."
991 EL = 0
992 IF EP > 100 THEN EP = EP - 1
00:EL = EL + 100: FOR X = 1 TO
58:M2(X) = M2(X) * 1.1: NEXT
: GOTO 992
993 EP = EP + EL
994 IF EL > 500 THEN FOR U = EL
TO 500 STEP - 100: FOR X =
1 TO 58:M2(X) = M2(X) / 1.1:
NEXT : NEXT
995 INPUT A$: HOME
997 A3 = 0:A4 = 0
998 IF A2 > ST * 2 THEN A4 = A2 -
ST * 2:A2 = ST * 2: PRINT "M
ORE THAN ";ST * 2;" MAGIC AR
ROWS WOULD","WEIGH YOU DOWN.
": FOR X = 1 TO 2000: NEXT X

```

```

999 IF A1 > ST * 2 THEN A3 = A1 -
ST * 2:A1 = ST * 2: PRINT "M
ORE THAN ";ST * 2;" ARROWS W
OULD","WEIGH YOU DOWN.": FOR
X = 1 TO 2000: NEXT X

```

Upon entering a new room, draw it with its monsters and treasures. If this is room one, give option to leave.

```

1000 HOME : IF B1 = 0 THEN B1 =
1: GOTO 1005
1001 IF RM < > 1 THEN 1005
1002 INPUT "DO YOU WANT TO LEAVE
THE DUNGEON? ";A$
1003 IF LEFT$(A$,1) = "Y" THEN
800
1004 HOME
1005 ON R1(RM) GOSUB 1000,11000
1010 IF T1(RM) > 0 THEN TX = FN
A(9) + 15:TY = FN A(6) + 9:
HTAB TX: VTAB TY: PRINT "*"
;
1015 X5 = 20:Y5 = 13
1020 IF I$ = "W" THEN Y5 = 22
1022 IF I$ = "X" THEN Y5 = 2
1024 IF I$ = "D" THEN X5 = 2
1026 IF I$ = "A" THEN X5 = 39
1028 HTAB X5: VTAB Y5: FLASH : PRINT
YY$;: NORMAL
1030 IF M2(RM) > = 1 THEN WX =
FN A(10) + 15:WY = FN A(6)
+ 9
1031 MS = MS(M1(RM)) / 10
1050 IF M2(RM) > = 1 THEN HTAB
WX: VTAB WY: PRINT M$(M1(RM)
);
Print player status, check for wandering monsters.
1055 HTAB 1: VTAB 2: PRINT "ARRO
WS:";A1;" "; HTAB 1: VTAB 1
: PRINT "M. ARROWS:";A2;" ";
1060 HTAB 1: VTAB 22: PRINT "ST=
";ST;" DX=";DX;: HTAB 1: VTAB
3: PRINT "WOUNDS:"; INT (W *
100 + .5);"%;" ";
1061 HTAB 1: VTAB 4: PRINT "ROOM
:";RM;" ";
1062 HTAB 33: VTAB 18: PRINT "PD
TIONS:"; HTAB 35: VTAB 19: PRINT
PT;" ";

```

continued on next page

continued from previous page

```
1063 HTAB 33: VTAB 20: PRINT "HO
LY M20"; HTAB 35: VTAB 21: PRINT
HW;" ";
1065 IF M2(RM) > = 1 THEN HTAB
28: VTAB 1: PRINT "MONSTER:"
;: HTAB 30: VTAB 2: PRINT MN
$(M1(RM));
1070 IF M2(RM) < 1 THEN HTAB 28
: VTAB 1: PRINT "
";: HTAB 30: VTAB 2: PRINT
" ";
1075 IF M2(RM) > 1 THEN HTAB 28
: VTAB 3: PRINT "NUMBER:"; INT
(M2(RM));" ";
1077 IF M2(RM) < = 1 THEN HTAB
28: VTAB 3: PRINT "
";
1080 HTAB 1: VTAB 18: PRINT "EX:
"; INT (EP);" ";: HTAB 35: VTAB
5: PRINT "W";: HTAB 34: VTAB
6: PRINT "A D";: HTAB 35: VTAB
7: PRINT "X";
1085 HTAB 1: VTAB 19: PRINT "GP:
";GP;" ";
1086 IF M2(RM) = 0 AND FN A(100
) = 1 THEN FOR X = 1 TO 10:
HTAB 5: VTAB 24: PRINT "WAN
DERING MONSTER!";: FOR Y = 1
TO 50: NEXT Y: HTAB 5: VTAB
24: PRINT "
";: FOR Y = 1 TO 50: NEXT Y
,X:M2(RM) = FN A(3):M1(RM) =
FN A(8): GOTO 1030

Accept a command from keyboard and
call appropriate subroutines.

1090 A$ = "": FOR X = 1 TO DX * 1
0 - EP: IF PEEK ( - 16384) >
128 THEN X = 9999
1091 NEXT X: IF X > 9000 THEN GET
A$
1093 IF T1(RM) > 0 THEN HTAB TX
: VTAB TY: PRINT "*";
1095 IF A$ = "" THEN A$ = I$: GOTO
1100
1097 I$ = A$
1100 IF I$ = "W" THEN GOSUB 151
00
1105 IF I$ = "X" THEN GOSUB 152
00
1110 IF I$ = "D" THEN GOSUB 153
00
1115 IF I$ = "A" THEN GOSUB 154
00
1120 IF I$ = "H" THEN I$ = "": IF
PT > 0 THEN PT = PT - 1:W =
1
```

```
1125 IF I$ = "M" AND A2 > 0 THEN
I$ = "":A2 = A2 - 1: GOSUB 1
5500
1130 IF I$ = "N" AND A1 > 0 THEN
I$ = "":A1 = A1 - 1: GOSUB 1
5600
1135 IF I$ = "F" THEN GOSUB 160
00
1140 IF I$ = "O" THEN GOSUB 170
00
1145 IF I$ = "T" AND HW > 0 THEN
I$ = "":HW = HW - 1: GOSUB 1
8000

If there is a monster in the room,
move it and let it attack.

1200 IF M2(RM) < 1 THEN 1030
1201 IF MS < = 0 THEN FOR X =
1 TO LEN (K$): HTAB WX: VTAB
WY: PRINT MID$( K$,X,1);: NEXT
X:M2(RM) = M2(RM) - 1:EP = E
P + MS(M1(RM)): GOTO 1030
1205 MX = SGN (X5 - WX)
1206 IF WX < X5 THEN MX = 1
1207 IF WX = X5 THEN MX = 0
1210 MY = SGN (Y5 - WY)
1215 HTAB WX: VTAB WY: PRINT " "
;
1220 IF SCR N( WX + MX - 1,WY *
2 - 2) + 16 * SCR N( WX + MX
- 1,WY * 2 - 1) = 160 THEN
WX = WX + MX
1225 IF SCR N( WX - 1,MY * 2 + W
Y * 2 - 2) + 16 * SCR N( WX -
1,MY * 2 + WY * 2 - 1) = 160
THEN WY = WY + MY
1230 IF ABS (WX - X5) > 1 OR ABS
(WY - Y5) > 1 THEN 1050
1235 X = RND (1): IF X > MS THEN
1050
1240 X = RND (1) * MS
1245 W = W - X: IF W < 0 THEN 500
0
1250 GOTO 1050

End-routine for the 'Great Dungeon
in the Sky' ending.

5000 FOR X = 1 TO LEN (K$): HTAB
X5: VTAB Y5: PRINT MID$( K$
,X,1);: NEXT X: FOR X = 1 TO
1000: NEXT X: HOME
5005 PRINT "WELCOME TO HEAVEN, "
;NM$;"!!!"
5010 PRINT "I HOPE YOU ENJOYED Y
OUR SHORT LIFETIME IN WHICH
YOU ACCUMULATED ";GP;" GOLD
"
```

```
5015 PRINT "AND ";EP;" EXPERIENC
E POINTS."
5020 PRINT : PRINT : INPUT "WOUL
D YOU LIKE TO BE REINCARNATE
D AS A NEW CHARACTER ?";A$:
IF LEFT$( A$,1) = "N" THEN
STOP
5025 RUN

Subroutine to draw a
passage/intersection.

10000 REM DRAW HALLWAY (TYPE #
1)
10002 INVERSE
10005 X1 = R2(RM,1)
10010 IF X1 > 0 THEN FOR X = 1 TO
8: HTAB 15: VTAB X: PRINT "
";: HTAB 26: PRINT " ";: NEXT
X
10012 IF X1 < = 0 THEN HTAB 16
: VTAB 9: PRINT " "
;
10015 X1 = R2(RM,2)
10020 IF X1 > 0 THEN FOR X = 16
TO 23: HTAB 15: VTAB X: PRINT
" ";: HTAB 26: PRINT " ";: NEXT
X
10022 IF X1 < = 0 THEN HTAB 16
: VTAB 16: PRINT "
";
10025 X1 = R2(RM,3)
10030 IF X1 > 0 THEN FOR X = 26
TO 40: HTAB X: VTAB 9: PRINT
" ";: HTAB X: VTAB 16: PRINT
" ";: NEXT X
10032 IF X1 < = 0 THEN FOR X =
9 TO 16: HTAB 26: VTAB X: PRINT
" ";: NEXT X
10035 X1 = R2(RM,4)
10040 IF X1 > 0 THEN FOR X = 1 TO
15: HTAB X: VTAB 9: PRINT "
";: HTAB X: VTAB 16: PRINT "
";: NEXT X
10042 IF X1 < = 0 THEN FOR X =
9 TO 16: HTAB 15: VTAB X: PRINT
" ";: NEXT X
10045 NORMAL : RETURN

Subroutine to draw a chamber/room.

11000 REM DRAW CHAMBER(TYPE#2)
11002 INVERSE
11005 HTAB 9: VTAB 5: PRINT "
";: HTAB 26: PRINT "
";
```

```

11010 HTAB 9: VTAB 20: PRINT "
      "; HTAB 26: PRINT "
      ";
11012 FOR X = 6 TO 9: HTAB 9: VTAB
      X: PRINT " "; NORMAL : PRINT
      " ";
      INVERSE : PRINT " "; NEXT X
11014 FOR X = 16 TO 19: HTAB 9: VTAB
      X: PRINT " "; NORMAL : PRINT
      " ";
      INVERSE : PRINT " "; NEXT X
11015 X1 = R2(RM,1)
11020 IF X1 > 0 THEN FOR X = 1 TO
      4: HTAB 15: VTAB X: PRINT "
      "; HTAB 26: PRINT " "; NEXT
      X
11022 IF X1 < = 0 THEN HTAB 16
      : VTAB 5: PRINT " "
      ;
11025 X1 = R2(RM,2)
11030 IF X1 > 0 THEN FOR X = 21
      TO 23: HTAB 15: VTAB X: PRINT
      " "; HTAB 26: PRINT " "; NEXT
      X
11032 IF X1 < = 0 THEN HTAB 16
      : VTAB 20: PRINT " "
      ;
11035 X1 = R2(RM,3)
11040 IF X1 > 0 THEN HTAB 32: VTAB
      9: PRINT " "; HTAB
      32: VTAB 16: PRINT " "
      ;
11042 IF X1 < = 0 THEN FOR X =
      10 TO 15: HTAB 32: VTAB X: PRINT
      " "; NEXT X
11045 X1 = R2(RM,4)
11050 IF X1 > 0 THEN HTAB 1: VTAB
      9: PRINT " "; HTAB
      1: VTAB 16: PRINT " "
      ;
11052 IF X1 < = 0 THEN FOR X =
      10 TO 15: HTAB 9: VTAB X: PRINT
      " "; NEXT X
11055 NORMAL : RETURN

```

Subroutines for moving player
around screen.

```

15100 IF Y5 = 1 THEN 15105
15102 M = SCR(N(X5 - 1, Y5 * 2 -
      4) + 16 * SCR(N(X5 - 1, Y5 *
      2 - 3) - 128: IF M < > 32 THEN
      RETURN
15105 HTAB X5: VTAB Y5: PRINT "
      ";
15110 Y5 = Y5 - 1: IF Y5 < 2 THEN
      RM = R2(RM,1): GOTO 1000

```

```

15120 HTAB X5: VTAB Y5: FLASH : PRINT
      YY$; NORMAL : RETURN
15200 IF Y5 = 22 THEN 15205
15202 M = SCR(N(X5 - 1, Y5 * 2) +
      16 * SCR(N(X5 - 1, Y5 * 2 +
      1) - 128: IF M < > 32 THEN
      RETURN
15205 HTAB X5: VTAB Y5: PRINT "
      ";
15210 Y5 = Y5 + 1: IF Y5 > 22 THEN
      RM = R2(RM,2): GOTO 1000
15220 GOTO 15120
15300 IF X5 > 37 THEN 15305
15302 M = SCR(N(X5, Y5 * 2 - 2) +
      16 * SCR(N(X5, Y5 * 2 - 1) -
      128: IF M < > 32 THEN RETURN
15303 M = SCR(N(X5 + 1, Y5 * 2 -
      2) + 16 * SCR(N(X5 + 1, Y5 *
      2 - 1) - 128: IF M < > 32 THEN
      RETURN
15305 HTAB X5: VTAB Y5: PRINT "
      ";
15310 X5 = X5 + 2: IF X5 > 39 THEN
      RM = R2(RM,3): GOTO 1000
15320 GOTO 15120
15400 IF X5 < = 2 THEN 15405
15402 M = SCR(N(X5 - 2, Y5 * 2 -
      2) + 16 * SCR(N(X5 - 2, Y5 *
      2 - 1) - 128: IF M < > 32 THEN
      RETURN
15403 M = SCR(N(X5 - 3, Y5 * 2 -
      2) + 16 * SCR(N(X5 - 3, Y5 *
      2 - 1) - 128: IF M < > 32 THEN
      RETURN
15405 HTAB X5: VTAB Y5: PRINT "
      ";
15410 X5 = X5 - 2: IF X5 < 1 THEN
      RM = R2(RM,4): GOTO 1000
15420 GOTO 15120

```

Normal arrow firing routine.

```

15500 GOSUB 15699
15505 X = RND(1) / 2: IF RC = 1
      THEN X = X - .1
15506 IF RC = 2 THEN X = X + .1
15507 X = X - (EP / 1000)
15510 X = X - .2
15511 X = X - (DX / 100)
15515 IF X > W THEN RETURN
15520 X = RND(1): IF RC = 1 THEN
      X = X + .2
15522 IF RC < > 1 THEN X = X +
      .1
15523 IF RC = 2 THEN X = X + .1
15525 MS = MS - X: RETURN
15599 RETURN

```

Magic arrow firing routine.

```

15600 GOSUB 15699
15601 IF M1(RM) = 8 THEN RETURN
15605 X = RND(1) / 2: IF RC = 1
      THEN X = X - .1
15606 X = X - (DX / 100)
15607 IF RC = 2 THEN X = X + .1
15608 X = X - (EP / 1000)
15610 IF X > W THEN RETURN
15620 X = RND(1): IF RC = 1 THEN
      X = X + .1
15621 IF RC = 2 THEN X = X - .1
15625 MS = MS - X: RETURN
15698 RETURN

```

Calculate monster range, aim, and
shoot arrow graphically.

```

15699 IF WX = 0 THEN WX = 31: IF
      WY = 0 THEN WY = 8
15700 X6 = X5: Y6 = Y5 - 1: X7 = WX
      : Y7 = WY - 1
15701 IF X6 = X7 THEN SL = 0: X8 =
      X7: X9 = X6
15702 IF X6 < > X7 THEN SL = (Y
      6 - Y7) / (X6 - X7): X8 = X6:
      X9 = X7
15703 GOTO 15708
15705 IF X6 > X7 THEN SL = (Y6 -
      Y7) / (X6 - X7): X8 = X6: X9 =
      X7
15706 IF X7 > X6 THEN SL = (Y7 -
      Y6) / (X7 - X6): X8 = X7: X9 =
      X6
15707 IF X7 = X6 THEN SL = 0: X8 =
      X7: X9 = X6
15708 Y8 = Y6: Y9 = Y7
15709 Y = Y8
15710 SL = SL * SGN(Y8 - Y9): IF
      Y6 < Y7 THEN SL = - SL
15711 IF X6 > X7 THEN SL = - SL
15712 FOR X = X8 TO X9 STEP SGN
      (X9 - X8 + .01): FOR XX = 1 TO
      20: NEXT XX
15713 IF Y > 23 OR Y < 1 OR X >
      40 OR X < 1 THEN NEXT X: GOTO
      15750
15715 IF SCR(N(X - 1, INT(Y) *
      2 - 2) + SCR(N(X - 1, INT(Y)
      * 2 - 1) * 16 = 32 THEN X
      9 = X - 1: GOTO 15750
15720 HTAB X: VTAB Y: PRINT "+";
      : Y = Y + SL: NEXT X
15750 Y = Y8: FOR X = X8 TO X9 STEP
      SGN(X9 - X8 + .01): HTAB X

```

continued on next page

continued from previous page

```
: VTAB Y: PRINT " ";Y = Y +
SL: NEXT X
15760 RETURN

Subroutine for close combat with a
monster.

16000 IF ABS (X5 - WX) > 1 OR ABS
(Y5 - WY) > 1 THEN RETURN
16001 IF M1(RM) = 8 THEN RETURN

16002 IF M1(RM) = 7 THEN RETURN

16003 IF M1(RM) = 6 THEN W = W -
.05
16005 X = RND (1): IF RC = 0 THEN
X = X - .1
16006 X = X - (DX / 100)
16007 IF RC = 2 THEN X = X - .3
16008 X = X - (EP / 1000)
16010 IF X > W THEN RETURN
16015 X = RND (1): IF RC = 0 THEN
X = X + .1
16016 X = X + (ST / 100)
16017 IF RC = 2 THEN X = X + .2
16020 MS = MS - X: RETURN

Subroutine for obtaining a
treasure.

17000 IF ABS (TX - X5) > 1 THEN
RETURN
17005 IF ABS (TY - Y5) > 1 THEN
RETURN
17010 HTAB TX: VTAB TY: PRINT "
";
17011 TX = 0:TY = 0
17015 HTAB 8: VTAB 24: PRINT T$(
T1(RM)); FOR X = 1 TO 1000:
NEXT X
17020 FOR X = 8 TO 35: HTAB X: VTAB
24: PRINT " "; NEXT X
17021 IF T1(RM) = 10 THEN PT = P
T + 1: GOTO 17026
17022 IF T1(RM) = 11 THEN A2 = A
2 + 10: GOTO 17026
17023 IF T1(RM) = 12 THEN A1 = A
1 + 10: GOTO 17026
17024 TS(T1(RM)) = TS(T1(RM)) + 1

17025 GP = GP + GP(T1(RM))
17026 EP = EP + EP(T1(RM)):T1(RM)
= 0
17030 RETURN
```

Subroutine to throw a flask of holy
water.

```
18000 M = M1(RM): IF M = 2 OR M =
5 OR M = 7 THEN RETURN
18005 GOSUB 15699: HTAB WX: VTAB
WY: PRINT " ";WA = WX:WB =
WY:WX = X5:WY = Y5: GOSUB 16
000:WX = WA:WY = WB: HTAB X5
: VTAB Y5: FLASH : PRINT YY$
:; NDRMAL : RETURN
```

Take care of 'end-of-quest'
procedures such as saving games
on tape or disk, and printing out
information on the player's
fighter.

```
20000 INPUT "WOULD YOU LIKE TO SE
E THE TREASURES THAT YOU RET
RIEVED FROM THE DUNGEON ?";A
$: IF LEFT$(A$,1) = "Y" THEN
FOR X = 1 TO 9: PRINT T$(X)
; TAB(30);TS(X): NEXT X
20010 FOR X = 1 TO 9:TS(X) = 0: NEXT
X
20011 A1 = A1 + A3:A2 = A2 + A4
20015 INPUT "WOULD YOU LIKE TO S
AVE THIS GAME ?";A$
20017 IF LEFT$(A$,1) < > "Y" THEN
20028
20018 INPUT "DISK OR CASSETTE ?"
;A$: IF LEFT$(A$,1) = "C" THEN
INPUT "HIT <RETURN> TO BEGI
N SAVE ";A$: PRINT : PRINT "
(WAIT FOR SIX BEEPS.)": STORE
R1: STORE M2: STORE T1: GOTO
20027
20019 IF LEFT$(A$,1) < > "D" THEN
20018
20020 INPUT "ENTER FILENAME >";A
$
20021 PRINT D$;"OPEN ";A$
20022 PRINT D$;"DELETE ";A$
20023 PRINT D$;"OPEN ";A$
20024 PRINT D$;"WRITE ";A$
20025 FOR X = 1 TO 58: PRINT M1(
X): PRINT M2(X): PRINT T1(X)
: NEXT X
20026 PRINT D$;"CLOSE ";A$
20027 PRINT "SAVE COMPLETE."
20028 INPUT "WOULD YOU LIKE TO S
TOP NOW ?";A$: IF LEFT$(A$
,1) < > "Y" THEN RETURN
20030 PRINT "OK. SO THAT YOU CA
N USE THIS CHARACTER AGAIN A
T A LATER TIME"
20035 PRINT "NAME: ";NM$;" RACE:
";: IF RC = 0 THEN PRINT "
```

HUMAN"

```
20036 IF RC = 1 THEN PRINT "ELF
"
20037 IF RC = 2 THEN PRINT "DWA
RF"
20040 PRINT "WOUNDS: ";W # 100;"
%"
20045 PRINT "HEALING POTIONS: ";
PT
20046 PRINT "HOLY WATER: ";HW
20050 PRINT "ARROWS: ";A1;" MA
GIC ARROWS: ";A2
20055 PRINT "GOLD: ";GP;" EXPE
RIENCE: ";EP
20060 PRINT "STRENGTH: ";ST;"
DEXTERITY: ";DX
20065 INPUT "WOULD YOU LIKE TO T
RY AGAIN AS A *NEW* CHARACT
ER ?";A$: IF LEFT$(A$,1) =
"Y" THEN RUN
20099 PRINT : PRINT "COME QUESTI
NG AGAIN SOMETIME!!!": END

Subroutine to create new fighter
characters.

21000 PRINT "OK, I'LL MAKE YOU O
NE.": FOR X = 1 TO 1000: NEXT
X
21005 GP = FN A(20) + 5:ST = FN
A(17) + 3:DX = FN A(17) + 3
:RC = FN A(3) - 1:A1 = 3:A2
= FN A(10):PT = FN A(3) +
1:HW = FN A(5):EP = 0:W = 1

21010 PRINT "STRENGTH: ";ST;"
DEXTERITY: ";DX
21015 PRINT "GOLD: ";GP;" HEALI
NG POTIONS: ";PT
21020 PRINT "HOLY WATER: ";HW;"
RACE: ";: IF RC = 0 THEN PRINT
"HUMAN"
21021 IF RC = 1 THEN PRINT "ELF
"
21022 IF RC = 2 THEN PRINT "DWA
RF"
21025 PRINT "ARROWS: ";A1;" MA
GIC ARROWS: ";A2
21030 INPUT "WHAT WILL YOU NAME
THIS CHARACTER ?";NM$:
PRINT "HAVE A FUN QUEST, ";
NM$;"!!!": FOR X = 1 TO 1000
: NEXT X: HOME
21040 RETURN
```

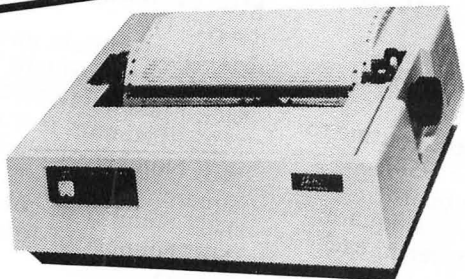
**AUGUST
SPECIALS!**
Direct from

TSE-HARDSIDE



LE SHHSTICK for the ATARI 800/400

Le Stick is the newest, most sophisticated joystick ever designed for your ATARI Home Computer or Video Game. Inside the grip are tiny incline switches which sense the movement of your hand and translate the information to your computer screen. Graphics programs and action games take on a new dimension. Large, conveniently located push button. Sensitive "squeeze switch" inside grip to freeze motion. Le Stick's one-handed operation gives you more control and added realism.
AUGUST 1 \$31.95
(This offer good until 9/15 only; no exceptions)



OKIDATA MICROLINE-82 PRINTER

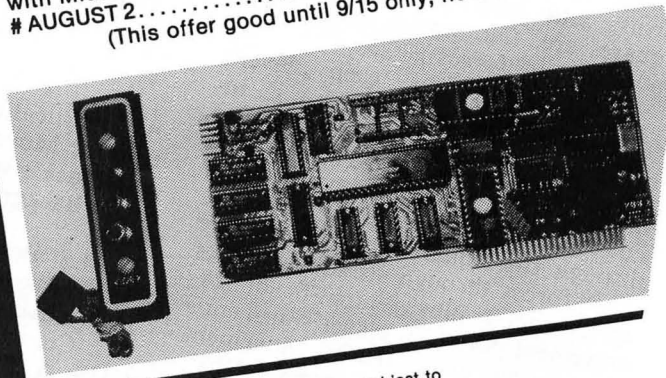
The MicroLine-82 is designed for applications that require greater speeds and more extensive forms controls. It prints bidirectionally at 80 cps and includes short line seeking logic that can increase throughput by 80% over equivalent unidirectional printers. Like the MicroLine-80, it is an 80-column unit with capability to produce 132 columns with condensed characters. Forms controls include vertical tab, top of form, and a vertical format unit that provides switch and program selection for up to ten form lengths.
AUGUST 3 \$529.00
(This offer good until 9/15 only; no exceptions)

T-TIMER CLOCK for TRS-80 Model-I

The T-TIMER uses a real-time clock/calendar chip, that will provide addressable I/O data for Seconds, Minutes, Hours, Day of the week, Day, Month, and Year. Program selectable 24/12 hour operation. Crystal controlled timebase. On-board battery backup (batteries not included), T-TIMER keeps operating even with the system off; battery life is approximately one year. Set Time Protect Switch to prevent accidental time changes.
AUGUST 4 \$69.95
(This offer good until 9/15 only; no exceptions)

VIDEX 80-COLUMN VIDEO DISPLAY BOARD for APPLE-II

This board permits 80 columns of text to be displayed giving your APPLE-II word processing capability. In addition, upper- and lower-case characters can be entered from the keyboard and displayed in a 7 x 9 dot matrix. Once the board is installed, it can be selected or deselected using the supplied manual switch plate (a \$19.95 value). An EPROM option is also available to drive the D.C. Hayes MicroModem. This board is completely compatible with MicroSoft's Z-80 SoftCard.
AUGUST 2 \$295.00
(This offer good until 9/15 only; no exceptions)



TERMS: Prices and specifications are subject to change. TSE HARDWARE accepts VISA & MASTERCARD, Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). TSE HARDWARE 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



Battlefield

by Joe Humphrey

Atari and S-80 translations by Jon Voskuil

“Battlefield” is an S-80, Apple, and Atari game requiring 16K RAM.

“Battlefield” is a game of strategy for two players, in which the object is to overpower the opponent’s forces and capture squares on the playing board. The players both start out with 40 squares on an eight-by-ten board, and alternate moving forces around the board to do battle with one another.

At the beginning of the first player’s turn, new forces are added to each of the squares along the right and left edges of the board. The number of new forces that each player receives depends on how much territory he owns, how many edge squares he owns, and how crowded his edge squares are. (No square may contain more than 99 forces.)

At the same time, each player’s Movement Allowance (M.A.) is increased by an amount proportional to the territory he owns. The M.A. limits the amount of movement on each turn. For example, moving five forces through two squares, and then three other forces through six squares, uses up 5-times-2 plus 3-times-6, or 28, movement points. A player’s turn ends when he has exhausted his Movement Allowance or when he presses “E”. (Unused M.A. is carried over to the next turn.)

Forces are moved using the number keys and four directional keys. The board display will show a pair of “#” symbols bracketing one of the squares; these constitute the “cursor”. The cursor can be moved up, left, right, or down by pressing I, J, K, or L (these keys form a diamond-shaped pattern for convenient right-hand use). To move forces, you position the cursor on a square occupied by your forces, type in the number of forces you want to move, and then use the directional keys to move them. To leave some forces in a given square, just type “0” or backspace to cancel the current number of forces being moved.

You can move your forces freely, in any direction, through your own territory. Once a group of forces crosses into enemy territory, however, they cannot move further during that turn.

When a square is thus occupied by forces from both sides, a pair of letters is shown in that square rather than a number of forces. These letters show the relative number of forces in that square: The pair “YB” would show very unequal force strengths, while the pair “MM” would show equal forces from both sides. At the end of each turn, battles are fought to the death in all such jointly-occupied squares, and the surviving forces of the winner then remain in the squares.

These battles for individual squares are usually, but not always, won by the side with more forces in the square. Assume that the forces in a square are 15 of side A and 10 of side B. In the first round of the battle, there will be a $10/(15+10)$, or 0.4, probability that

one of A’s forces will be destroyed; at the same time there will be a $15/(15+10)$, or 0.6, probability that one of B’s forces will be destroyed. At the end of the first round, then, the remaining forces could be 15:10, 14:10, 15:9, or 14:9. Assuming that the actual results are 15:9, the second round would be fought with new probabilities of $9/(15+9)$ and $15/(15+9)$, or 0.31 and 0.69 respectively. Rounds continue in this fashion until one player’s forces are completely destroyed in that square.

If you should want to reduce the size of the “Battlefield” playing board, to shorten and simplify the game, it’s easily done by changing line 70. NC is the number of columns and NR the number of rows.

VARIABLES

AR: Vertical tab or PRINT@ location for Movement Allowance.
B\$: Black’s square marker.
BC\$: Background character (CU\$ or SP\$).
BELL\$: Beeps the speaker.
BL: Index to black player (= 0).
BS\$: Backspace character, CHR\$(8).
CC: Cursor column.
CH\$: A character.
CL\$: Atari clear-screen character, CHR\$(125).
CLEOS: Address of Apple routine to clear to end of screen (= -958).
COL: A column on the board.
CP: Controlling player in a square.
CP(*, *): Controlling player in each square of array.
CU\$: Cursor character, “#”.
D: Sound distortion number.
DC: Change in column CC (-1, 0, +1).
DC(*): Change in column for each player.
DF: Digit flag; = 1 if a digit was just typed.
DR: Change in row CR.
EC(*): End-of-line column for each player.
F: Boolean “false” value (= 0).
FR: Vertical tab or PRINT@ position for number of forces.
HC: Half of NC.
IPL: Index to a player.

IV: Value for printing normal (= 0) or inverse (= 128) character.
KB: Address of Apple keyboard buffer.
LC(*): Label column for each player.
LP: Timing-loop counter.
MA(*): Movement Allowance for each player.
MC: Maximum column index (= NC-1).
MF(*): Maximum number of incoming forces per square for each player.
MG: Maximum value of NG allowed.
MR: Maximum row index (= NR-1).
NB: Number of black forces in a square.
NC: Number of columns on board.
NF: Number of forces to add to a square.
NG: Number of forces in a moving group.
NR: Number of rows on board.
NS(*): Number of squares owned by each player.
NW: Number of white forces in a square.
PC: Player’s name column.
PF: Previous value of DF.
PL: Index to current player.
PN\$, PN1\$, PN2\$, PN\$(*): Strings used for players’ names.
PR: Player’s-name row.
ROW: A row on the board.

S: Flag for sound generation.
 SF: A square's number of forces.
 SF(*,*), SF(*,*,*): Square's forces in each square, for each player.
 SP\$: Space character.
 SR: Vertical tab or PRINT@ position for squares owned.

T: Boolean "true" value (= 1).
 TF(*): Total forces for each player.
 V: Sound volume value.
 W\$: White's square marker.
 WH: Index to white player (= 1).
 WP\$: Winning player's name.
 X\$: Temporary and utility string.

APPLE VERSION

Main program control routine.

```

60 CLEAR
70 NC = 8:NR = 10
80 GOSUB 1000
90 GOSUB 2000
100 GOSUB 3000
110 GOSUB 4000
120 GOSUB 5000
130 GOSUB 6000
140 GOSUB 7000
150 GOSUB 8000
160 HOME : END

Print instructions.

1000 TEXT : HOME
1010 PRINT "BATTLEFIELD: A TERRI
TORTIAL GAME,"
1020 PRINT "WRITTEN BY JOE HUMPH
REY, NOV 1980."
1030 PRINT
1040 PRINT " TWO PLAYERS, BLAC
K AND WHITE, ARE ON"
1050 PRINT "A ";NC;"X";NR;" BOAR
D. A SQUARE OWNED BY BLACK"

1060 PRINT "IS SHOWN WITH THE NU
MBER OF BLACK"
1070 PRINT "FORCES IN THAT SQUAR
E, AGAINST A BLACK"
1080 PRINT "BACKGROUND; A WHITE-
OWNED SQUARE HAS A"
1090 PRINT "WHITE BACKGROUND. A
SQUARE CONTAINING"
1100 PRINT "BOTH SIDES' FORCES I
S SHOWN WITH"
1110 PRINT "LETTERS REPRESENTING
THE RELATIVE FORCE"

```

```

1120 PRINT "SIZES; AT THE END OF
EACH TURN, A"
1130 PRINT "BATTLE IS FOUGHT (TO
THE DEATH) IN EACH"
1140 PRINT "OF THESE SQUARES, TO
DETERMINE WHO OWNS"
1150 PRINT "IT. THE FIRST PLAYE
R TO OWN ALL ";NC;" NR
1160 PRINT "SQUARES WINS."
1170 PRINT " AT THE BEGINNING
OF BLACK'S TURN,"
1180 PRINT "NEW FORCES COME ONTO
EACH SIDE, AND"
1190 PRINT "EACH SIDE'S MOVEMENT
ALLOWANCE (M.A.)"
1200 PRINT "IS INCREASED (THE M.
A. DETERMINES HOW"
1210 PRINT "MANY FORCES MAY BE M
OVED AROUND)."
1220 RETURN

```

Initialize variables.

```

2000 BL = 0:WH = 1
2010 F = 0:T = 1
2020 MC = NC - 1:MR = NR - 1
2030 DIM DC(1),EC(1),LC(1),PC(1)

2040 DIM MA(1),MF(1),NS(1),PN$(1
),TF(1)
2050 DIM CP(MC,MR),SF(1,MC,MR)
2060 LC(BL) = 1:DC(BL) = 16:EC(BL
) = 21
2070 LC(WH) = LC(BL) + 20:DC(WH) =
DC(BL) + 20:EC(WH) = EC(BL) +
19
2080 PR = 16:FR = PR + 2:SR = FR +
2:AR = SR + 2
2090 BS$ = CHR$(8):BELL$ = CHR$(
7)
2100 SP$ = " ";CU$ = "#"
2110 CLEOS = - 958:KB = - 16384

```

```

2120 HC = INT (NC / 2)
2130 FOR ROW = 0 TO MR
2140 FOR COL = 0 TO HC - 1:CP(CO
L,ROW) = BL: NEXT COL
2150 FOR COL = HC TO MC:CP(COL,R
OW) = WH: NEXT COL
2160 NEXT ROW
2170 NS(BL) = NR * HC:NS(WH) = NR
* NC - NS(BL)
2180 RETURN

```

Input players' names.

```

3000 VTAB 23: INPUT "BLACK PLAYE
R'S NAME? ";PN$
3010 IF PN$ = "" THEN PN$ = "BLA
CK"
3020 PN$(BL) = LEFT$(PN$,19):PC
(BL) = 11 - LEN (PN$(BL)) /
2
3030 VTAB 23: CALL CLEOS: INPUT
"WHITE PLAYER'S NAME? ";PN$
3040 IF PN$ = "" THEN PN$ = "WHI
TE"
3050 PN$(WH) = LEFT$(PN$,19):PC
(WH) = 31 - LEN (PN$(WH)) /
2
3060 RETURN

```

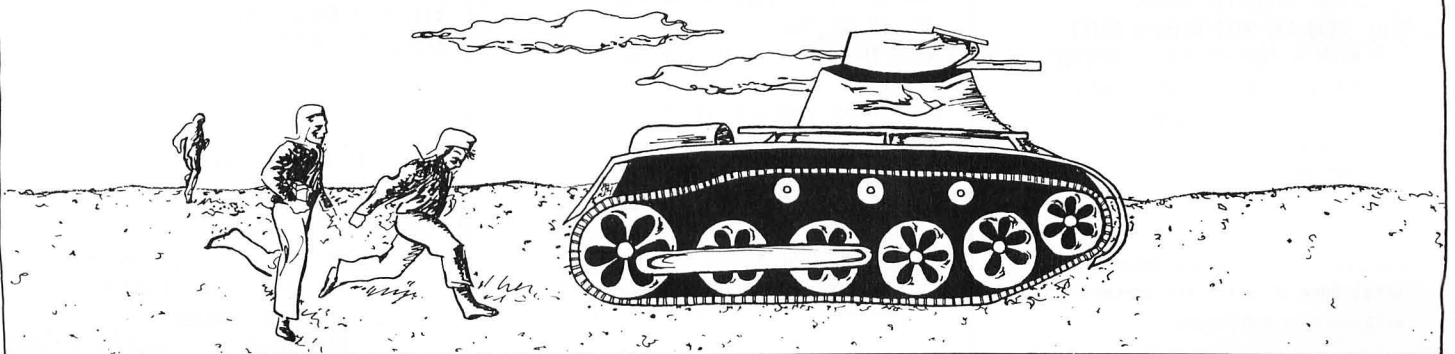
Draw playing field and display player data.

```

4000 BC$ = SP$
4010 VTAB 4: HTAB 1: CALL CLEOS
4020 FOR ROW = 0 TO MR: FOR COL =
0 TO MC
4030 GOSUB 9000
4040 NEXT COL: NEXT ROW
4050 FOR IPL = BL TO WH
4060 VTAB PR
4070 HTAB LC(IPL): PRINT TAB( P
C(IPL));PN$(IPL); TAB( EC(IPL
L)); PRINT
4080 HTAB LC(IPL): PRINT TAB( E
C(IPL)); PRINT
4090 HTAB LC(IPL): PRINT "TOTAL
FORCES :"; TAB( DC(IPL));TF(
IPL); TAB( EC(IPL)); PRINT
4100 HTAB (LC(IPL)); PRINT TAB(
EC(IPL)); PRINT

```

continued on next page





continued from previous page

```

4110 HTAB LC(IPL); PRINT "SQUARE
      S OWNED:"; TAB( DC(IPL)); NS(
      IPL); TAB( EC(IPL)); PRINT
4120 HTAB LC(IPL); PRINT TAB( E
      C(IPL)); PRINT
4130 HTAB LC(IPL); PRINT "MOVEME
      NT LEFT:"; TAB( DC(IPL)); MA(
      IPL); TAB( EC(IPL))
4140 INVERSE : NEXT IPL
4150 NORMAL
4160 RETURN
  
```

Begin Black's turn. Add new forces to edge squares and increase movement allowances.

```

5000 BC% = SP%
5010 FOR IPL = BL TO WH
5020 MF(IPL) = INT (NS(IPL) / NR
      ) + 1
5030 NEXT IPL
5040 FOR ROW = 0 TO MR: FOR COL =
      0 TO MC STEP MC
5050 CP = CP(COL,ROW):SF = SF(CP,
      COL,ROW)
5060 NF = MF(CP): IF SF + NF > 99
      THEN NF = 99 - SF
5070 IF NF > 0 THEN SF(CP,COL,RO
      W) = SF + NF: GOSUB 9000
5080 TF(CP) = TF(CP) + NF:MA(CP) =
      MA(CP) + MF(CP)
5090 NEXT COL: NEXT ROW
5100 VTAB FR: HTAB DC(BL): PRINT
      TF(BL); TAB( EC(BL)); INVERSE
      : HTAB DC(WH): PRINT TF(WH);
      TAB( EC(WH)); NORMAL
5110 VTAB AR: HTAB DC(BL): PRINT
      MA(BL); TAB( EC(BL)); INVERSE
      : HTAB DC(WH): PRINT MA(WH);
      TAB( EC(WH)); NORMAL
5120 PL = BL:CC = 0
5130 RETURN
  
```

Execute player's moves; move men until done or until his movement allowance is exhausted.

```

6000 CR = INT (NR / 2)
6010 NG = 0
6020 PF = F
6030 VTAB PR: HTAB PC(PL): FLASH
      : PRINT PN$(PL);
6040 BC% = CU%:COL = CC:ROW = CR:
      GOSUB 9000
6050 GOSUB 11000
6060 MG = SF(PL,CC,CR)
6070 IF MA(PL) = 0 THEN 6120
6080 GOSUB 10000
6090 IF CH% = "H" THEN 6030
6100 IF CH% = BS% OR DF THEN 605
      0
6110 IF CH% < > "E" AND CH% < >
      "S" THEN 6060
6120 BC% = SP%:COL = CC:ROW = CR:
      GOSUB 9000
6130 NG = 0: GOSUB 11000
6140 NORMAL : IF PL = WH THEN INVERSE
6150 VTAB PR: HTAB PC(PL): PRINT
      PN$(PL); NORMAL
6160 RETURN

Fight battles to the death in all
squares which contain both players'
forces.

7000 IF NS(PL) = 0 THEN 7170
7010 FOR ROW = 0 TO MR: FOR COL =
      0 TO MC
7020 CP = CP(COL,ROW): IF CP = PL
      OR SF(PL,COL,ROW) = 0 THEN
      7160
7030 NS(CP) = NS(CP) - 1
7040 NB = SF(BL,COL,ROW):NW = SF(
      WH,COL,ROW)
7050 IF NW < > NB THEN CP(COL,R
      OW) = (NW > NB)
7060 IF NW * NB = 0 THEN 7130
7070 SF = NB + NW
7080 IF NB < SF * RND (1) THEN
      NB = NB - 1:TF(BL) = TF(BL) -
      1
7090 IF NW < SF * RND (1) THEN
      NW = NW - 1:TF(WH) = TF(WH) -
      1
  
```

```

7100 VTAB FR: HTAB DC(BL): PRINT
      TF(BL); TAB( EC(BL)); HTAB
      DC(WH): INVERSE : PRINT TF(W
      H); TAB( EC(WH)); NORMAL
7110 SF(BL,COL,ROW) = NB:SF(WH,CO
      L,ROW) = NW
7120 GOSUB 9000: GOTO 7050
7130 GOSUB 9000
7140 CP = CP(COL,ROW):NS(CP) = NS
      (CP) + 1
7150 VTAB SR: HTAB DC(BL): PRINT
      NS(BL); TAB( EC(BL)); HTAB
      DC(WH): INVERSE : PRINT NS(W
      H); TAB( EC(WH)); NORMAL
7160 NEXT COL: NEXT ROW
7170 RETURN
  
```

End of a turn. Check for end of game; if not, transfer control to other player.

```

8000 IF NS(BL) * NS(WH) > 0 THEN
      8100
8020 WP% = PN$(BL): IF NS(BL) = 0
      THEN WP% = PN$(WH)
8030 VTAB 23: HTAB 1: CALL CLEOS

8040 HTAB 18 - LEN (WP%) / 2: FLASH
      : PRINT WP%;" WINS"
8050 NORMAL : PRINT "ANOTHER GAM
      E (Y/N)?" ;
8060 GET CH%
8070 IF CH% = "N" THEN 8130
8080 IF CH% < > "Y" THEN 8060
8090 POP : GOTO 60
8100 IF PL = WH THEN POP : GOTO
      120
8110 PL = WH:CC = MC
8120 POP : GOTO 130
8130 RETURN
  
```

Print the contents of position (COL,ROW) to the screen.

```

9000 CP = CP(COL,ROW)
9010 NORMAL : IF (BC% = SP% AND
      CP = WH) OR (BC% = CU% AND P
      L = WH) THEN INVERSE
  
```

```

9020 VTAB ROW + 4: HTAB 4 * COL +
6: PRINT BC$;
9030 VTAB ROW + 4: HTAB 4 * COL +
3: PRINT BC$;
9040 NB = SF(BL,COL,ROW):NW = SF(
WH,COL,ROW):SF = NB + NW
9050 IF SF( NOT CP,COL,ROW) = 0 THEN
9090
9060 NORMAL : PRINT CHR$( NB /
SF * 25 + ASC ("A"));
9070 INVERSE : PRINT CHR$( NW /
SF * 25 + ASC ("A"));
9080 GOTO 9120
9090 NORMAL : IF CP = WH THEN INVERSE
9100 IF SF < 10 THEN PRINT " ";

9110 PRINT SF;
9120 NORMAL
9130 RETURN

```

Input and execute a command.

```

10000 IF PEEK (KB) < 128 THEN 1
0000
10010 GET CH$
10020 POKE - 16368,0
10030 DF = (CH$ > = "0" AND CH$ <
= "9"):DC = 0:DR = 0
10040 IF DF AND NOT PF THEN NG =
0
10050 PF = DF
10060 IF DF THEN NG = 10 * NG +
VAL (CH$): GOTO 10160
10070 IF CH$ = BS$ THEN NG = INT
(NG / 10):PF = 1: GOTO 10160

10080 IF CH$ = "E" THEN 10160
10090 IF CH$ = "H" THEN GOSUB 1
2000: GOSUB 4000: GOTO 10160

10100 IF CH$ = "I" THEN DR = -
1: GOSUB 13000: GOTO 10160
10110 IF CH$ = "J" THEN DC = -
1: GOSUB 13000: GOTO 10160
10120 IF CH$ = "K" THEN DC = 1: GOSUB
13000: GOTO 10160
10130 IF CH$ = "M" THEN DR = 1: GOSUB
13000: GOTO 10160

```

```

10140 IF CH$ = "S" THEN NS(PL) =
0: GOTO 10160
10150 PRINT BELL$;
10160 RETURN

```

Adjust the number of forces being moved, and update the screen display.

```

11000 IF CP(CC,CR) < > PL THEN
NG = 0
11010 IF NG > MG THEN NG = MG
11020 IF NG > MA(PL) THEN NG = M
A(PL)
11030 VTAB 24: HTAB 1: NORMAL
11040 IF NG = 0 THEN PRINT TAB(
12);"<TYPE ";: INVERSE : PRINT
"H";: NORMAL : PRINT " FOR H
ELP";: GOTO 11080
11050 PRINT TAB( 11);"<MOVING "
;NG;" FORCE";
11060 IF NG > 1 THEN PRINT "S";

11070 PRINT ">";
11080 CALL CLEOS
11090 RETURN

```

Print a list of commands (in response to <H>elp request).

```

12000 VTAB 4: HTAB 1: CALL CLEOS

12010 PRINT "COMMAND:","DESCRIPT
ION: "
12020 PRINT
12030 PRINT "<-","DELETE LAST DI
GIT OF","NUMBER; "
12040 PRINT "A NUMBER","(E.G. 23
) THE NUMBER OF","FORCES TO
MOVE; "
12050 PRINT "E","END TURN (THE T
URN ALSO)","ENDS WHEN THE MO
VEMENT","ALLOWANCE REACHES
0); "

```

```

12060 PRINT "H","PRINT THIS TEXT
; "
12070 PRINT "I","MOVE SOME FORCE
S UP; "
12080 PRINT "J","MOVE SOME FORCE
S LEFT; "
12090 PRINT "K","MOVE SOME FORCE
S RIGHT; "
12100 PRINT "M","MOVE SOME FORCE
S DOWN; "
12110 PRINT "S","SURRENDER TO TH
E OTHER","SIDE."
12120 VTAB 24: PRINT TAB( 7);"<
HIT ANY KEY TO RESUME GAME>"
;
12130 GET CH$:CH$ = "H"
12140 RETURN

```

Move cursor and forces, and update screen display.

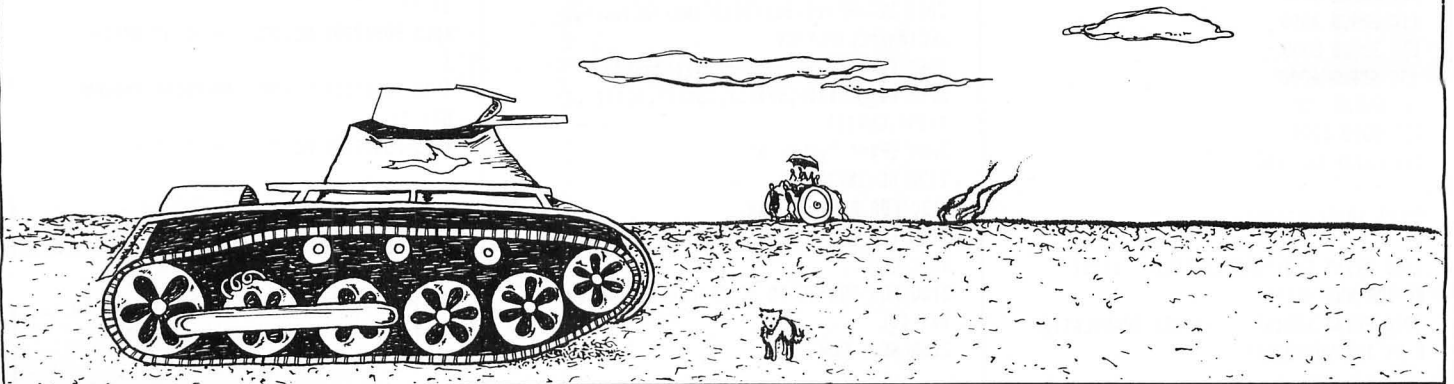
```

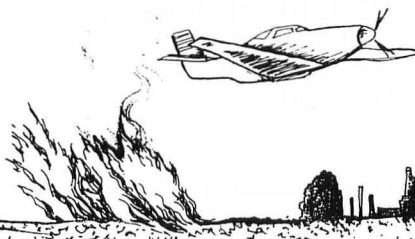
13000 SF(PL,CC,CR) = SF(PL,CC,CR)
- NG
13010 BC$ = SP$:COL = CC:ROW = CR
: GOSUB 9000
13020 CC = CC + DC:CC = CC + (CC <
0) - (CC > MC)
13030 CR = CR + DR:CR = CR + (CR <
0) - (CR > MR)
13040 FS = SF(PL,CC,CR): IF FS +
NG > 99 THEN SF(PL,COL,ROW) =
SF(PL,COL,ROW) + NG - (99 -
FS):NG = 99 - FS: GOSUB 9000

13050 SF(PL,CC,CR) = FS + NG
13060 IF CC = COL AND CR = ROW THEN
PRINT BELL$;:NG = 0
13070 BC$ = CU$:COL = CC:ROW = CR
: GOSUB 9000
13080 MA(PL) = MA(PL) - NG: GOSUB
11000
13090 NORMAL : IF PL = WH THEN INVERSE
13100 VTAB AR: HTAB DC(PL): PRINT
MA(PL); TAB( EC(PL));
13110 NORMAL
13120 RETURN

```

continued on next page





continued from previous page

ATARI VERSION

10 GOTO 70

Subroutine to print a normal or inverse character.

```
20 FOR I=1 TO LEN(X%):PRINT CHR$(ASC(X
%(I,I))+IV);NEXT I:RETURN
```

Subroutine to print the contents of position (COL,ROW) to the screen.

```
30 CP=CP(COL,ROW):IV=0:IF (BC%=SP% AND
CP=WH) OR (BC%=CU% AND PL=WH) THEN IV
=128
```

```
32 IF S THEN SOUND 0,RND(1)*20+20,D,V
34 POSITION 4*COL+6,ROW+2:X%=BC%:GOSUB
20:SOUND 0,0,0,0:POSITION 4*COL+3,ROW
+2:GOSUB 20
```

```
36 NB=SF(COL,ROW):NW=SF(COL+NC,ROW):SF
=NB+NW:IF SF(COL+(1-CP)*NC,ROW)=0 THEN
40
```

```
38 IV=0:X%=CHR$(NB/SF*25+65):GOSUB 20:
IV=128:X%=CHR$(NW/SF*25+65):GOSUB 20:G
OTO 46
```

```
40 IV=0:IF CP=WH THEN IV=128
```

```
42 IF SF<10 THEN X%=" ":X%(2)=STR$(SF)
:GOSUB 20:GOTO 46
```

```
44 X%=STR$(SF):GOSUB 20
```

```
46 IV=0:RETURN
```

Main program control routine.

```
70 NC=8:NR=10
```

```
75 DIM CL$(1):CL%=CHR$(125)
```

```
78 POKE 752,1
```

```
80 GOSUB 1000
```

```
90 GOSUB 2000
```

```
100 GOSUB 3000
```

```
110 GOSUB 4000
```

```
120 GOSUB 5000
```

```
130 GOSUB 6000
```

```
140 GOSUB 7000
```

```
150 GOTO 8000
```

```
160 PRINT CL$:END
```

Print instructions.

```
1000 PRINT CL$;"BATTLEFIELD *****
BY JOE HUMPHREY"
```

```
1010 PRINT :PRINT " (ATARI TRANSLATIO
N BY JON VOSKUIL)"
```

```
1030 PRINT
```

```
1040 PRINT " TWO PLAYERS, BLACK AND
WHITE, ARE ON A ";NC;" BY ";NR;" BOAR
D. A SQUARE OWNED"
```

```
1050 PRINT "BY BLACK IS SHOWN WITH THE
NUMBER OF FORCES IN THAT SQUARE AGAI
NST A BLACK"
```

```
1060 PRINT "BACKGROUND; A WHITE-OWNED
SQUARE HAS A WHITE BACKGROUND."
```

```
1070 PRINT " SQUARES CONTAINING BOT
H SIDES' FORCES ARE SHOWN WITH LETT
ERS REPRE--"
```

```
1080 PRINT "SENTING THE RELATIVE FORCE
SIZES. AT THE END OF EACH TURN, BATT
LES ARE"
```

```
1090 PRINT "FOUGHT IN EACH OF THESE SQ
UARES, TO DETERMINE WHO OWNS THEM.
THE FIRST"
```

```
1100 PRINT "PLAYER TO OWN ALL THE SQUA
RES WINS."
```

```
1110 PRINT " AT THE BEGINNING OF EA
CH ROUND, NEW FORCES ARE ADDED TO EA
CH SIDE."
```

```
1120 PRINT "AND EACH SIDE'S MOVEMENT A
LLOWANCE IS INCREASED."
```

```
1130 RETURN
```

Initialize variables.

```
2000 BL=0:WH=1
```

```
2010 F=0:T=1
```

```
2020 MC=NC-1:MR=NR-1
```

```
2030 DIM DC(1),EC(1),LC(1),PC(1)
```

```
2040 DIM MA(1),MF(1),NS(1),TF(1),PN1%(
20),PN2%(20),WP%(15)
```

```
2050 DIM CP(MC,MR),SF(NC*2,MR)
```

```
2055 FOR I=0 TO 1:MA(I)=0:MF(I)=0:TF(I
)=0:NEXT I
```

```
2060 LC(BL)=4:DC(BL)=13:EC(BL)=21
```

```
2070 LC(WH)=LC(BL)+20:DC(WH)=DC(BL)+20
:EC(WH)=EC(BL)+19
```

```
2080 PR=15:FR=PR+2:SR=FR+2:AR=SR+2
```

```
2090 DIM BS%(1),SP%(1),CU%(1),BC%(1),X
%(20),CH%(1)
```

```
2100 SP%=" ":CU%="#"
```

```
2120 HC=INT(NC/2)
```

```
2130 FOR ROW=0 TO MR
```

```
2140 FOR COL=0 TO HC-1:CP(COL,ROW)=BL:
NEXT COL
```

```
2150 FOR COL=HC TO MC:CP(COL,ROW)=WH:N
EXT COL
```

```
2160 NEXT ROW
```

```
2170 NS(BL)=NR*HC:NS(WH)=NR*NC-NS(BL)
```

```
2175 FOR COL=0 TO NC*2:FOR ROW=0 TO MR
:SF(COL,ROW)=0:NEXT ROW:NEXT COL
```

```
2180 OPEN #2,4,0,"K:"
```

```
2190 RETURN
```

Input players' names.

```
3000 POKE 752,0:POSITION 2,22:PRINT "B
LACK PLAYER'S NAME: ";:INPUT PN1%
```

```
3010 IF PN1%="" THEN PN1%="BLACK"
```

```
3020 IF LEN(PN1%)>15 THEN PN1%=PN1%(1,
15)
```

```
3025 PC(BL)=9-LEN(PN1%)/2
```

```
3030 POSITION 2,23:PRINT "WHITE PLAYER
'S NAME: ";:INPUT PN2%
```

```
3040 IF PN2%="" THEN PN2%="WHITE"
```

```
3050 IF LEN(PN2%)>15 THEN PN2%=PN2%(1,
15)
```

```
3055 PC(WH)=29-LEN(PN2%)/2
```

```
3060 POKE 752,1:RETURN
```

Draw playing field and display player data.

```
4000 BC%=SP%
```

```
4010 PRINT CL%
```

```
4020 FOR ROW=0 TO MR:FOR COL=0 TO MC
```

```
4030 GOSUB 30
```

```
4040 NEXT COL:NEXT ROW
```

```
4050 FOR IPL=BL TO WH
```

```
4070 POSITION PC(IPL),PR:IF IPL=BL THE
N PRINT PN1%;
```

```
4075 IF IPL=WH THEN PRINT PN2%;
```

```
4090 POSITION LC(IPL),FR:PRINT "FORCES
:";
```

```
4095 POSITION DC(IPL),FR:PRINT TF(IPL)
;
```

```
4110 POSITION LC(IPL),SR:PRINT "SQUARE
S:";
```

```
4115 POSITION DC(IPL),SR:PRINT NS(IPL)
;
```

```
4130 POSITION LC(IPL),AR:PRINT "MOVEME
NT:";
```

```
4135 POSITION DC(IPL),AR:PRINT MA(IPL)
;
```

```
;
```

```
4140 NEXT IPL
```

```
4160 RETURN
```

Begin Black's turn. Add new forces to edge squares and increase movement allowances.

```

5000 BC$=SP$
5005 S=1:D=10:V=8
5010 FOR IPL=BL TO WH
5020 MF(IPL)=INT(NS(IPL)/NR)+1
5030 NEXT IPL
5040 FOR ROW=0 TO MR:FOR COL=0 TO MC S
TEP MC
5050 CP=CP(COL,ROW):SF=SF(COL+CP*NC,RO
W)
5060 NF=MF(CP):IF SF+NF>99 THEN NF=99-
SF
5070 IF NF>0 THEN SF(COL+CP*NC,ROW)=SF
+NF:GOSUB 30
5080 TF(CP)=TF(CP)+NF:MA(CP)=MA(CP)+MF
(CP)
5090 NEXT COL:NEXT ROW
5100 POSITION DC(BL),FR:PRINT TF(BL);"
";:POSITION DC(WH),FR:PRINT TF(WH);"
";
5110 POSITION DC(BL),AR:PRINT MA(BL);"
";:POSITION DC(WH),AR:PRINT MA(WH);"
";
5120 PL=BL:CC=0
5125 S=0
5130 RETURN

```

Execute player's moves; move men until done or until his movement allowance is exhausted.

```

6000 CR=INT(NR/2)
6010 NG=0
6020 PF=F
6030 POSITION PC(PL),PR:IV=128:X$=PN1$
:IF PL=WH THEN X$=PN2$
6035 GOSUB 20
6040 BC$=CU$:COL=CC:ROW=CR:GOSUB 30
6050 GOSUB 11000
6060 MG=SF(CC+PL*NC,CR)
6070 IF MA(PL)=0 THEN 6120
6080 GOSUB 10000
6090 IF CH$="H" THEN 6030
6100 IF CH$=BS$ OR DF THEN 6050
6110 IF CH$<>"E" AND CH$<>"S" THEN 606
0
6120 BC$=SP$:COL=CC:ROW=CR:GOSUB 30
6130 NG=0:GOSUB 11000
6150 IV=0:POSITION PC(PL),PR:X$=PN1$:I
F PL=WH THEN X$=PN2$
6155 GOSUB 20
6160 IV=0:RETURN

```

Fight battles to the death in all squares which contain both players' forces.

```

7000 IF NS(PL)=0 THEN 7170
7005 S=1:D=4:V=12
7010 FOR ROW=0 TO MR:FOR COL=0 TO MC
7020 CP=CP(COL,ROW):IF CP=PL OR SF(COL
+PL*NC,ROW)=0 THEN 7160
7030 NS(CP)=NS(CP)-1
7040 NB=SF(COL+BL*NC,ROW):NW=SF(COL+WH
*NC,ROW)
7050 IF NW<>NB THEN CP(COL,ROW)=(NW)NB
)
7060 IF NW*NB=0 THEN 7130
7070 SF=NB+NW
7080 IF NB<SF*RND(1) THEN NB=NB-1:TF(B
L)=TF(BL)-1
7090 IF NW<SF*RND(1) THEN NW=NW-1:TF(W
H)=TF(WH)-1
7100 POSITION DC(BL),FR:PRINT TF(BL);"
";:POSITION DC(WH),FR:PRINT TF(WH);"
";
7110 SF(COL+BL*NC,ROW)=NB:SF(COL+WH*NC
,ROW)=NW
7120 GOSUB 30:GOTO 7050
7130 GOSUB 30
7140 CP=CP(COL,ROW):NS(CP)=NS(CP)+1
7150 POSITION DC(BL),SR:PRINT NS(BL);"
";:POSITION DC(WH),SR:PRINT NS(WH);"
";
7160 NEXT COL:NEXT ROW
7165 S=0
7170 RETURN

```

End of a turn. Check for end of game; if not, transfer control to other player.

```

8000 IF NS(BL)*NS(WH)>0 THEN 8100
8020 WP$=PN1$:IF NS(BL)=0 THEN WP$=PN2
$
8025 POSITION 11,23:PRINT "
";
8030 FOR Z=1 TO 10:POSITION 2,21
8035 SOUND 0,50,10,10
8040 PRINT "
";:FOR ZZ=1 TO 10:NEXT ZZ
8045 SOUND 0,100,10,10
8050 POSITION 16-LEN(WP$)/2,21:PRINT W
P$;" WINS!!!";:FOR ZZ=1 TO 20:NEXT ZZ:

```

```

NEXT Z
8055 SOUND 0,0,0,0
8060 POKE 752,0:POSITION 14,23:PRINT "
ANOTHER GAME";:INPUT X$:IF X$="" THEN
8060
8070 IF X$(1,1)="N" THEN 160
8090 RUN
8100 IF PL=WH THEN 120
8110 PL=WH:CC=MC
8120 GOTO 130

```

Input and execute a command.

```

10000 GET #2,CH
10010 CH$=CHR$(CH)
10030 DF=(CH$)="0" AND CH$<="9":DC=0:
DR=0
10040 IF DF AND NOT PF THEN NG=0
10050 PF=DF
10060 IF DF THEN NG=10*NG+VAL(CH$):GOT
O 10160
10070 IF CH$=BS$ THEN NG=INT(NG/10):PF
=1:GOTO 10160
10080 IF CH$="E" THEN 10160
10090 IF CH$="H" THEN GOSUB 12000:GOSU
B 4000:GOTO 10160
10100 IF CH$="I" THEN DR=-1:GOTO 10150
10110 IF CH$="J" THEN DC=-1:GOTO 10150
10120 IF CH$="K" THEN DC=1:GOTO 10150
10130 IF CH$="M" THEN DR=1:GOTO 10150
10140 IF CH$="S" THEN NS(PL)=0:GOTO 10
160
10150 GOSUB 13000
10160 RETURN

```

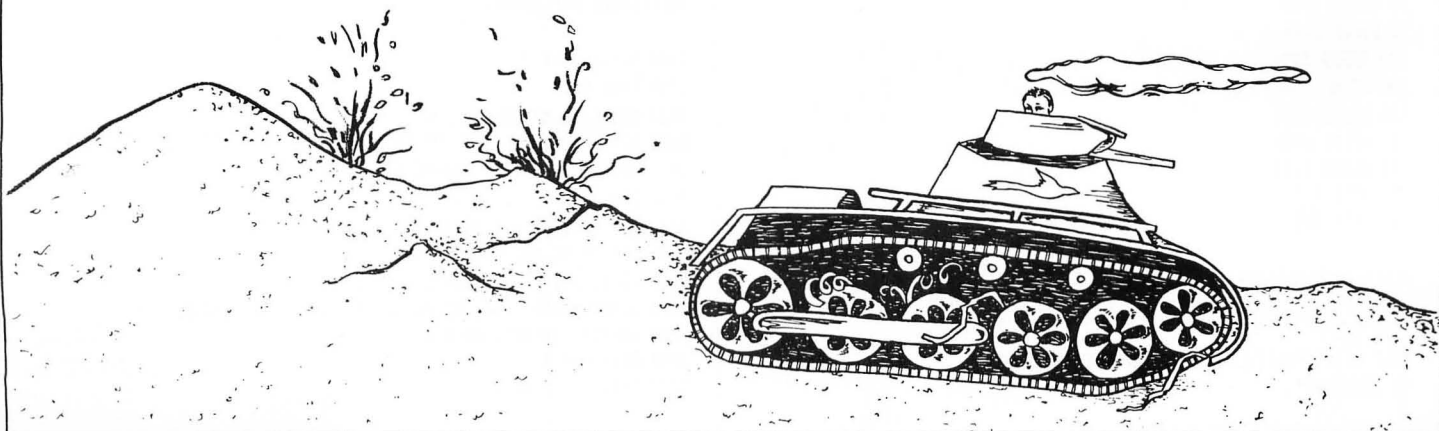
Adjust the number of forces being moved, and update the screen display.

```

11000 IF CP(CC,CR)<>PL THEN NG
11010 IF NG>MG THEN NG=MG
11020 IF NG>MA(PL) THEN NG=MA(PL)
11030 POSITION 11,23
11040 IF NG=0 THEN PRINT "<TYPE H FOR
HELP>";:GOTO 11080
11050 PRINT "<MOVING ";NG;" FORCE";
11060 IF NG>1 THEN PRINT "S";
11070 PRINT ">";
11080 PRINT " ";
11090 RETURN

```

continued on next page



continued from previous page

Print a list of commands (in response to <H>help request).

```
12000 PRINT CL$:PRINT "COMMAND:","  
DESCRIPTION:"
```

```
12010 PRINT :PRINT "A NUMBER","NUMBER  
OF FORCES TO MOVE."
```

```
12020 PRINT :PRINT "BACKSPC","DELETE L  
AST DIGIT OF NUMBER."
```

```
12030 PRINT " E","END TURN BEFORE MO  
VEMENT=0."
```

```
12040 PRINT :PRINT " H","PRINT THIS  
COMMAND LIST."
```

```
12050 PRINT :PRINT " I","MOVE SOME F  
ORCES UP."
```

```
12060 PRINT :PRINT " J","MOVE SOME F  
ORCES LEFT."
```

```
12070 PRINT :PRINT " K","MOVE SOME F  
ORCES RIGHT."
```

```
12080 PRINT :PRINT " M","MOVE SOME F  
ORCES DOWN."
```

```
12090 PRINT :PRINT " S","SURRENDER T  
O OTHER SIDE."
```

```
12100 PRINT :PRINT :PRINT " <HIT ANY  
KEY TO RETURN TO GAME>";
```

```
12110 GET #2,X
```

```
12120 RETURN
```

Move cursor and forces, and update screen display.

```
13000 SF(CC+PL*NC,CR)=SF(CC+PL*NC,CR)-  
NG
```

```
13010 BC*=SP$:COL=CC:ROW=CR:GOSUB 30
```

```
13020 CC=CC+DC:CC=CC+(CC<0)-(CC)MC)
```

```
13030 CR=CR+DR:CR=CR+(CR<0)-(CR)MR)
```

```
13040 FS=SF(CC+PL*NC,CR):IF FS+NG>99 T
```

```
HEN SF(COL+PL*NC,ROW)=SF(COL+PL*NC,ROW  
) +NG-(99-FS):NG=99-FS:GOSUB 30
```

```
13050 SF(CC+PL*NC,CR)=FS+NG
```

```
13060 IF CC=COL AND CR=ROW THEN NG=0
```

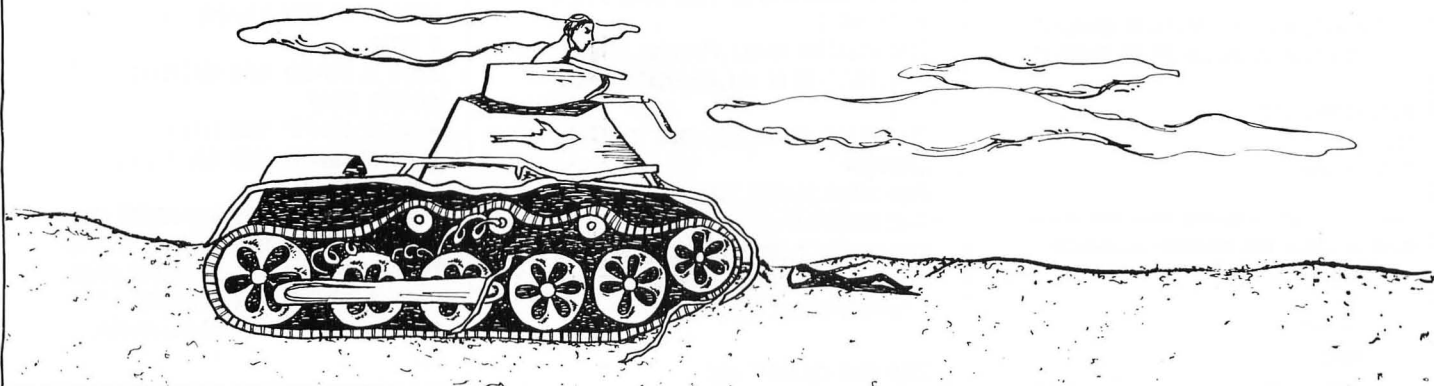
```
13070 BC*=CU$:COL=CC:ROW=CR:GOSUB 30
```

```
13080 MA(PL)=MA(PL)-NG:GOSUB 11000
```

```
13090 POSITION DC(PL),AR:PRINT MA(PL);
```

```
" ";
```

```
13120 RETURN
```



S-80 VERSION

```
10 GOTO 60
```

Subroutine to print contents of position (COL,ROW) to screen.

```
20 CP=CP(COL,ROW):PRINT@ PAX(COL,ROW),BC$:P$(CP):;NB=SF(BL,CO  
L,ROW):NW=SF(WH,COL,ROW):SF=NB+NW:IF SF(1-CP,COL,ROW)=0 THEN  
30
```

```
25 PRINT CHR$(NB/SF*25+65);CHR$(NW/SF*25+65):;GOTO 35
```

```
30 PRINT RIGHT$(STR$(SF),2);
```

```
35 PRINT P$(CP);BC$:;RETURN
```

Main program control routine.

```
60 CLEAR 200
```

```
65 B$=CHR$(138):W$="-"
```

```
70 NC=8:NR=10
```

```
80 GOSUB 1000
```

```
90 GOSUB 2000
```

```
100 GOSUB 3000
```

```
110 GOSUB 4000
```

```
120 GOSUB 5000
```

```
130 GOSUB 6000
```

```
140 GOSUB 7000
```

```
150 GOTO 8000
```

```
160 CLS:END
```

Print instructions.

```
1000 CLS
```

```
1010 PRINT "BATTLEFIELD: A TERRITORIAL GAME  
DE HUMPHREY"
```

BY J

```
1020 PRINT TAB(15) "(S-80 TRANSLATION BY JON VOSKUIL)":PRINT  
1040 PRINT" TWO PLAYERS CONTROL FORCES ON A";NC;"BY";NR;"PLAY  
ING FIELD."
```

```
1050 PRINT"SQUARES OWNED BY ONE SIDE ARE SHOWN WITH THE NUMBER O  
F FORCES","BETWEEN VERTICAL BARS (";CHR$(138);" ";CHR$(138);");  
SQUARES OWNED BY THE OTHER SIDE"
```

```
1060 PRINT"ARE SHOWN WITH THE NUMBER OF FORCES BETWEEN HYPHENS (  
- -)."," SQUARES CONTAINING FORCES OF BOTH SIDES ARE SHOWN W  
ITH"
```

```
1070 PRINT"LETTERS REPRESENTING THE RELATIVE FORCE SIZES; AT THE  
END OF","EACH TURN, BATTLES ARE FOUGHT IN THESE SQUARES TO DETE  
RMIINE"
```

```
1080 PRINT"WHO OWNS THEM. THE FIRST PLAYER TO OWN ALL THE SQUAR  
ES WINS."
```

```
1090 PRINT" AT THE BEGINNING OF EACH ROUND, NEW FORCES ARE AD  
DED TO EACHSIDE, AND EACH SIDE'S MOVEMENT ALLOWANCE IS INCREASED  
."
```

```
1110 RETURN
```

Initialize variables.

```
2000 BL=0:WH=1
```

```
2010 F=0:T=1
```

```
2020 MC=NC-1:MR=NR-1
```

```
2025 DIM PAX(MC,MR):FOR COL=0 TO MC:FOR ROW=0 TO MR:PAX(COL,R  
OW)=ROW*64+COL*8+(10-NC)*3+1:NEXT:NEXT
```

```
2030 DIM DC(1),EC(1),LC(1),PC(1)
```

```
2040 DIM MA(1),MF(1),NS(1),PN$(1),TF(1)
```

```
2050 DIM CP(MC,MR),SF(1,MC,MR)
```

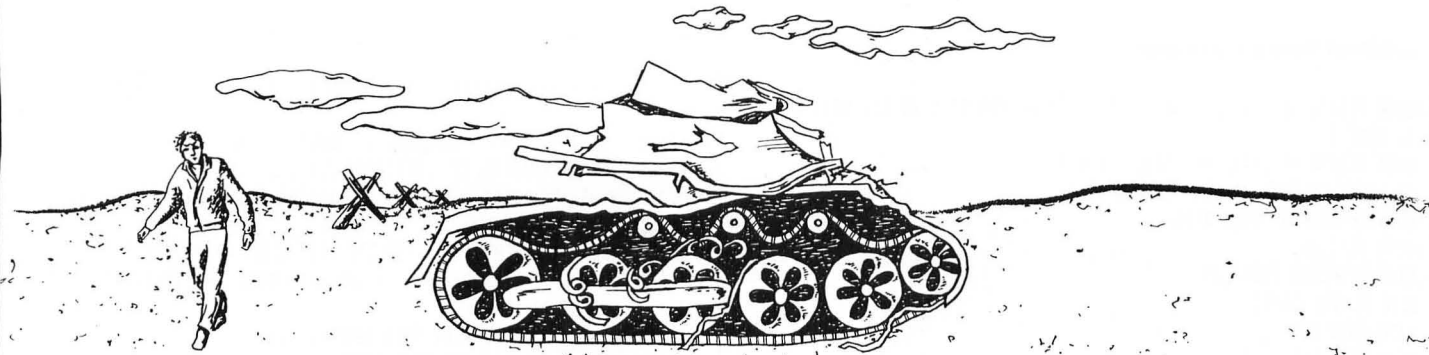
```
2060 LC(BL)=1:DC(BL)=16:EC(BL)=21
```

```
2070 LC(WH)=LC(BL)+20:DC(WH)=DC(BL)+20:EC(WH)=EC(BL)+19
```

```
2080 FR=839:SR=855:AR=914
```

```
2090 BS$=CHR$(8)
```

```
2100 SP$=" ":CU$=""
```

```

2110 DIM P$(1): P$(BL)="-": P$(WH)=CHR$(138)
2120 HC=INT(NC/2)
2130 FOR ROW=0 TO MR: FOR COL=0 TO HC-1: CP(COL,ROW)=BL: NEXT: F
OR COL=HC TO MC: CP(COL,ROW)=WH: NEXT: NEXT
2170 NS(BL)=NR*HC: NS(WH)=NR*NC-NS(BL)
2190 RETURN

```

Input players' names.

```

3000 PRINT @900,"PLAYER 1'S NAME: ";: INPUT PN$
3010 IF PN$="" THEN 3000
3020 IF LEN(PN$)>10 THEN PN$=LEFT$(PN$,10)
3025 PN$(BL)=PN$: PN$=""
3030 PRINT @900, STRING$(63,32);: PRINT @900, "PLAYER 2'S NAME:
";: INPUT PN$
3040 IF PN$="" THEN 3030
3050 IF LEN(PN$)>10 THEN PN$=LEFT$(PN$,10)
3055 PN$(WH)=PN$
3060 RETURN

```

Draw playing field and display player data.

```

4000 BC$=SP$
4010 CLS
4020 FOR ROW=0 TO MR: FOR COL=0 TO MC: GOSUB 20: NEXT: NEXT
4050 PRINT@ 704, STRING$(64,176);: FOR I=0 TO 128 STEP 64: PRINT
@ 799+I, CHR$(149);: NEXT I
4060 FOR IPL=BL TO WH: PC=IPL*32
4065 L=14-INT(LEN(PN$(IPL))/2)
4070 PRINT@ 769+PC, STRING$(L,95); PN$(IPL); STRING$(L,95);
4080 PRINT@ 832+PC, "FORCES:      SQUARES:";
4090 PRINT@ 900+PC, "MOVEMENT LEFT:";
4100 PRINT@ FR+PC, TF(IPL);
4110 PRINT@ SR+PC, NS(IPL);
4120 PRINT@ AR+PC, MA(IPL);
4130 NEXT IPL: RETURN

```

Begin black's turn. Add forces and increase Movement Allowances.

```

5000 BC$=SP$
5010 FOR IPL=BL TO WH
5020 MF(IPL)=INT(NS(IPL)/NR)+1
5030 NEXT IPL
5040 FOR ROW=0 TO MR: FOR COL=0 TO MC STEP MC
5050 CP=CP(COL,ROW): SF=SF(CP,COL,ROW)
5060 NF=MF(CP): IF SF+NF>99 THEN NF=99-SF
5070 IF NF>0 THEN SF(CP,COL,ROW)=SF+NF: GOSUB 20
5080 TF(CP)=TF(CP)+NF: MA(CP)=MA(CP)+MF(CP)
5090 NEXT COL,ROW
5100 FOR IPL=BL TO WH
5110 PRINT@ FR+IPL*32, TF(IPL);" ";

```

```

5120 PRINT@ AR+IPL*32, MA(IPL);" ";
5130 NEXT IPL: PL=BL: CC=0
5140 RETURN

```

Execute player's moves until done or until Movement Allowance is depleted.

```

6000 CR=INT(NR/2)
6010 NG=0
6020 PF=F
6030 GOSUB 14000
6040 BC$=CU$: COL=CC: ROW=CR: GOSUB 20
6050 GOSUB 11000
6060 MG=SF(PL,CC,CR)
6070 IF MA(PL)=0 THEN 6120
6080 GOSUB 10000
6090 IF CH$="H" THEN 6030
6100 IF CH$=BS$ OR DF THEN 6050
6110 IF CH$<>"E" AND CH$<>"S" THEN 6060
6120 BC$=SP$: COL=CC: ROW=CR: GOSUB 20
6130 NG=0:GOSUB 11000
6140 PRINT@ 960, STRING$(63,32);
6160 RETURN

```

Fight battles to the death in disputed squares.

```

7000 IF NS(PL)=0 THEN 7170
7010 FOR ROW=0 TO MR: FOR COL=0 TO MC
7020 CP=CP(COL,ROW): IF CP=PL OR SF(PL,COL,ROW)=0 THEN 7160
7030 NS(CP)=NS(CP)-1
7040 NB=SF(BL,COL,ROW): NW=SF(WH,COL,ROW)
7050 IF NW>NB THEN CP(COL,ROW)=- (NW>NB)
7060 IF NW*NB=0 THEN 7130
7070 SF=NB+NW
7080 IF NB<SF*2 THEN NB=NB-1: TF(BL)=TF(BL)-1
7090 IF NW<SF*2 THEN NW=NW-1: TF(WH)=TF(WH)-1
7100 FOR ZP=BL TO WH: PRINT@ FR+32*ZP, TF(ZP);: NEXT ZP
7110 SF(BL,COL,ROW)=NB: SF(WH,COL,ROW)=NW
7120 GOSUB 20: GOTO 7050
7130 GOSUB 20
7140 CP=CP(COL,ROW): NS(CP)=NS(CP)+1
7150 FOR ZP=BL TO WH: PRINT@ SR+32*ZP, NS(ZP);: NEXT ZP
7160 NEXT: NEXT
7170 RETURN

```

End of turn; check for end of game.

```

8000 IF NS(BL)*NS(WH)>0 THEN 8100
8020 WP$=PN$(BL): IF NS(BL)=0 THEN WP$=PN$(WH)
8025 FOR I=LEN(WP$)-1 TO 1 STEP -1: WP$=LEFT$(WP$,I) + " " + RIG
HT$(WP$,LEN(WP$)-I): NEXT I
8030 FOR Z=1 TO 30: PRINT@ 960, STRING$(63,32);: FOR ZZ=1 TO 10:
NEXT ZZ

```

continued on next page

continued from previous page

```
8040 PRINT@ 960, WP$;" W I N S ! ! !"; FOR ZZ=1 TO 15: NEXT Z
Z: NEXT Z
8050 PRINT@ 990, "ANOTHER GAME? (Y/N)";
8060 CH#=INKEY$; IF CH#="" THEN 8060
8070 IF CH#="N" THEN 8130
8090 GOTO 60
8100 IF PL=WH THEN 120
8110 PL=WH: CC=MC
8120 GOTO 130
8130 GOTO 160
```

Input and execute a command.

```
10000 CH#=INKEY$; IF CH#="" THEN 10000
10030 DF= (CH#)="0" AND CH#<="9": DC=0: DR=0
10040 IF DF AND NOT PF THEN NG=0
10050 PF=DF
10060 IF DF THEN NG=10*NG+VAL(CH#): GOTO 10160
10070 IF CH#=BS$ THEN NG=INT(NG/10): PF=1: GOTO 10160
10080 IF CH#="E" THEN 10160
10090 IF CH#="H" THEN GOSUB 12000: GOSUB 4000: GOTO 10160
10100 IF CH#="I" THEN DR=-1: GOTO 10150
10110 IF CH#="J" THEN DC=-1: GOTO 10150
10120 IF CH#="K" THEN DC=1: GOTO 10150
10130 IF CH#="M" THEN DR=1: GOTO 10150
10140 IF CH#="S" THEN NG(PL)=0: GOTO 10160
10150 GOSUB 13000
10160 RETURN
```

Adjust the number of forces being moved, and update the screen.

```
11000 IF CP(CC,CR)<>PL THEN NG=0
11010 IF NG>MG THEN NG=MG
11020 IF NG>MA(PL) THEN NG=MA(PL)
11030 PRINT@ 982,;
11040 IF NG=0 THEN PRINT "<TYPE H FOR HELP>"; GOTO 11080
11050 PRINT "<MOVING>;NG;"FORCE";
11060 IF NG>1 THEN PRINT "S";
11070 PRINT">";
```

```
11080 PRINT " ";
11090 RETURN
```

Print a list of commands, in response to request for <H>elp.

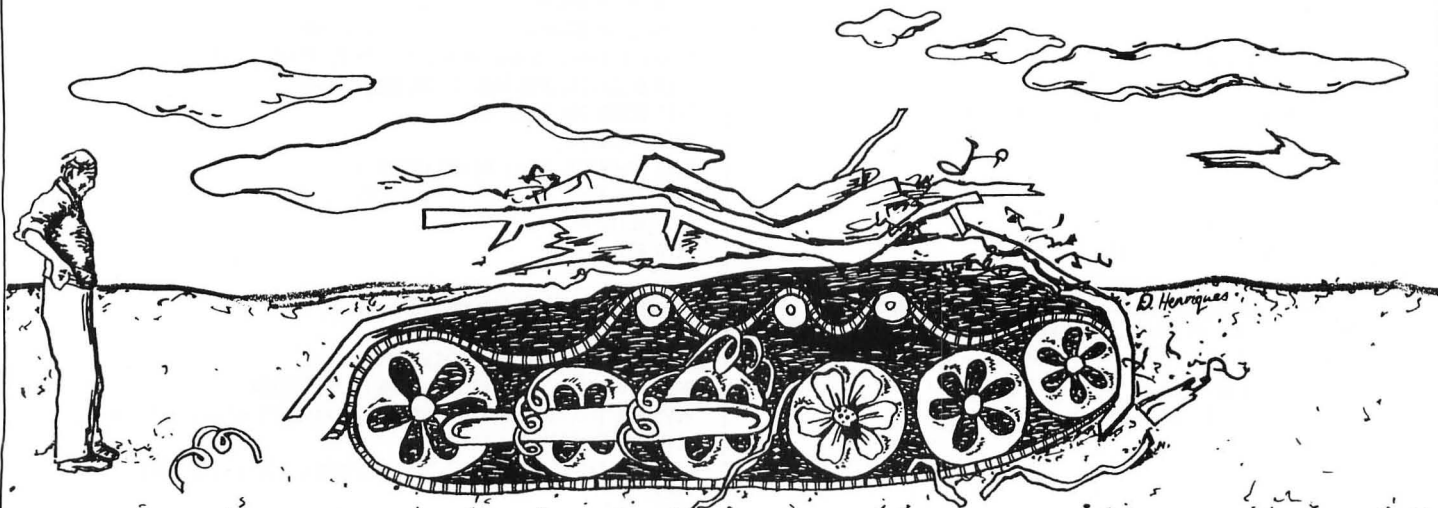
```
12000 CLS: PRINT@ 64, "COMMAND:", "DESCRIPTION:"
12005 PRINT STRING$(8,131),STRING$(12,131)
12010 PRINT: PRINT "A NUMBER", "THE NUMBER OF FORCES TO MOVE (E.
6., 23).";
12020 PRINT " ";CHR$(93), "DELETE LAST DIGIT OF NUMBER."
12030 PRINT" E", "END TURN. (TURN ALSO ENDS WHEN MOVEMENT",,
" ALLOWANCE IS ZERO.)";
12040 PRINT " H", "PRINT THIS COMMAND LIST."
12050 PRINT " I", "MOVE SOME FORCES UP."
12060 PRINT " J", "MOVE SOME FORCES LEFT."
12070 PRINT " K", "MOVE SOME FORCES RIGHT."
12080 PRINT " M", "MOVE SOME FORCES DOWN."
12090 PRINT " S", "SURRENDER TO THE OTHER SIDE."
12100 PRINT@ 980, "<HIT ANY KEY TO RETURN TO GAME>";
12110 CH#=INKEY$; IF CH#="" THEN 12110
12120 CH#="H": RETURN
```

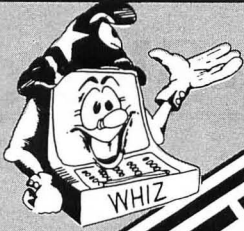
Move cursor and forces, and update the screen.

```
13000 SF(PL,CC,CR)=SF(PL,CC,CR)-NG
13010 BC#=SP$: COL=CC: ROW=CR: GOSUB 20
13020 CC=CC+DC: CC=CC-(CC<0)+(CC>MC)
13030 CR=CR+DR: CR=CR-(CR<0)+(CR>MR)
13040 FS=SF(PL,CC,CR): IF FS+NG>99 THEN SF(PL,COL,ROW)=SF(PL,COL
,ROW)+NG-(99-FS): NG=99-FS: GOSUB 20
13050 SF(PL,CC,CR)=FS+NG
13060 IF CC=COL AND CR=ROW THEN NG=0
13070 BC#=CU$: COL=CC: ROW=CR: GOSUB 20
13080 MA(PL)=MA(PL)-NG: GOSUB 11000
13100 PRINT@ AR+PL*32, MA(PL);
13120 RETURN
```

Routine to flash a player's name.

```
14000 PRINT@ 960, STRING$(63,32);
14010 FOR Z=1 TO 4: PRINT@ 960+45*PL, STRING$(18,32);
14020 FOR ZZ=1 TO 15: NEXT ZZ
14030 PRINT@ 960+45*PL, PN$(PL);"S TURN";
14040 FOR ZZ=1 TO 20: NEXT ZZ
14050 NEXT Z: RETURN
```





ADVENTURE INTERNATIONAL

PRO-PIX '81

By Talley-Ho Software

PRO-FOOTBALL-PIX, or PRO-PIX, is the culmination of over five years of development and use of a utility program to track the progress of the 28 professional U. S. football teams during the regular 224-game (16 weeks and 14 games per week) season. PRO-PIX made its public debut in 1980 under a slightly different name after extensive testing and was very successful, receiving many plaudits from users, and requests for a 1981 version. PRO-PIX is basically an updated version for 1981, with several subtle changes in presentation format. The prediction data has been modified slightly and information is included herein for updating the program for successive seasons. PRO-PIX is designed for use on APPLE II PLUS or APPLE II, or ATARI 400/800 computers with at least 32K of memory. It will operate with data-handling to either Disk or Tape.

Features of PRO-PIX may be summarized as follows:

- List SCHEDULES by team or week of interest.
- List SCORES of all games played, by team or week.
- List current STANDINGS in division.
- Show PREDICTIONS for games to be played, by team or week.
- PRINT any screen that displays data.
- UPDATE the program by entering weekly scores.
- Set up a new SEASON.



PRO-PIX '81

By Talley-Ho Software

TRS-80 TAPE Model 1&3	010-0127	\$19.95
TRS-80 DISK Model 1	012-0127	\$24.95

ALSO AVAILABLE FOR

APPLE TAPE TO DISK	041-0127	\$19.95
ATARI 400/800 TAPE TO DISK	051-0127	\$19.95

ADVENTURE 11 & 12 ARE HERE!

CALL OUR TOLL-FREE NUMBER FOR ORDERING INFORMATION!

INTERACTIVE FICTION

BY ROBERT LaFORE

FOURTH IN THE SERIES

HIS MAJESTY'S SHIP "IMPETUOUS"

WHAT IS IT?

Interactive Fiction is story-telling using a computer, so that you, the reader, can actually take part in the story instead of merely reading.

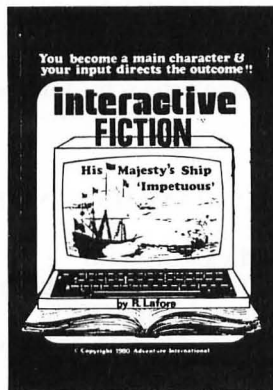
HOW DOES IT WORK?

The computer sets the scene with a fictional situation, which you can read from the CRT. Then, you become a character in the story: when it's your turn to speak you type in your response. The dialogue of the other characters and even the plot will depend on what you say.

IS IT A GAME?

No. In a game the situation is rigidly defined and you can select from only a limited number of responses. But in Interactive Fiction you can say anything you like to the other characters. (Of course if your response is too bizarre they may not understand you.)

His Majesty's Ship "Impetuous" — You are the Captain, Horatio Hornblower, back in the days when His Majesty's Navy ruled the seven seas. Pirates, plunder, fame and fortune await the intrepid captain. If you have ever enjoyed books about the sea, now is your chance to take the helm and find out what this life was really like.



TRS-80 Model 1 32K DISK

His Majesty's Ship "Impetuous"

012-0077 \$19.95

ALSO AVAILABLE FOR

APPLE 2 - 48K DISK

(Applesoft in ROM required)

His Majesty's Ship "Impetuous"

042-0077 \$19.95

TRS-80™

The *Classic* game you've heard so much about!



A Division of Scott Adams, Inc.
Adventure International • Box 3435 • Longwood, FL 32750
CATALOG AND TOLL-FREE ORDER NUMBER: (800) 327-7172
 IN FLORIDA (305) 862-8817
 SHIPPING AND HANDLING CHARGES ARE EXTRA
 PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE





Compu-Sketch

by Roger W. Robitaille, Sr.

“Compu-Sketch” is an S-80 graphics program requiring 16K of memory.

“Compu-Sketch” is a nifty little program that will transform anybody into a computer artist. The screen becomes your canvas, the keyboard your brush. And after you complete your masterpiece, you can electronically store and preserve it for the world to see.

This program was designed to be simple to use, and provides plenty of prompting to the user through the use of menus and menu-like displays. The first menu you will encounter allows you to choose from the five major parts of the program: the Draw routine, which is explained in detail below; the Image Save and Image Read routines, which will move a memory full of designs to and from either cassette or disk; a printer output routine which will print your artwork on your graphic printer; and a routine that allows you to review, delete and modify any of the images in memory.

The draw mode is where most of the interaction with the program takes place, and this is where the actual drawing is done. You will be presented with a list of options at the bottom of the screen, and a flashing cursor in the middle of the screen. Imagine this point as being the tip of your paintbrush, which you will move, pick up, and set down to draw.

On the bottom right of the screen are two displays which give you some important information. REPT tells you whether the automatic repeat is on or off. When it is on, depressing one of the arrow keys will cause the cursor to continue to move until it is released. When it is off, a keystroke is needed for every space which the dot is to move. The other message here is FUNC, which tells you what function the program is in: MOVE means that movement of the cursor will not alter the screen display, SET signifies that moving the cursor will leave a line of points, and RESET will cause the cursor to continually erase wherever it moves. One last function, TYPE, tells the computer to display all keyboard input on the graphics screen instead of interpreting it as a command. Hit the

CLEAR key to exit this mode and return to MOVE.

Another important display is CURSOR. Within each double-wide graphics block, there are three positions which can be separately accessed. The CURSOR display shows which of those three vertically-stacked positions is currently being addressed.

The legal commands while in the Draw routine are:

G: Turns the automatic repeat on or off.

F: Allows you to change the function (SET, RESET, MOVE or TYPE).

I: Allows you to save the current image in memory.

S: Sets (turns on) the point at the current cursor location.

R: Resets (turns off) the point at the current cursor location.

C: Allows you to enter a single character on the screen by typing either the character or the character's ASCII value. (Illegal values will be ignored.)

M: Returns you to the main menu at the start of the program, leaving all stored images intact. Remember to store the image on the screen before doing this, or it will be lost.

The arrow keys: Used to move the cursor around the screen in the desired direction.

PROGRAMMER'S NOTES

Softside's original S-80 screen drawing program was published in its fourth issue (January 1979). It required less than 1K of code and did a very creditable job for its time. It was structured around SET/RESET graphics. One of its main drawbacks was linked to the nature of the S-80 graphics themselves. A single pixel (the smallest possible graphics element) is not as wide as it is tall; in fact, it's just about twice as tall as it is wide. Lines that are a single pixel wide appear twice as thick when drawn across as when drawn down. It made for some awkward-looking drawings, to be sure.

The solution was to devise a method to support a cursor that was two pixels wide by one pixel tall. This approach produces a nearly equal vertical versus horizontal line thickness. This simple concept, however, has some rather

complicated consequences. Let me explain.

The principles of the 64 graphics characters supported by the S-80 have been explained again and again in **SoftSide** and elsewhere. There are six pixels that occupy the same area of the screen that would otherwise be a letter. Since this program uses them in horizontal pairs, there are only three of the new double pixels in that same area. And, whereas the six normal rectangular pixels can be arranged in 64 combinations, this program uses only eight of them: Graphic 128 (all off), 131 (upper two on), 140 (middle two on), 143 (upper and middle pairs on), 179 (lower pair on), 188 (middle and lower pairs on), and 191 (all on).

Much of the logic of this program is based on knowing that adding or subtracting the proper offset values will result in turning on or off the proper pixel pair. The top pair can be turned on or off by adding or subtracting the value 3, the middle pair 12, and the bottom pair 48. There is more to it of course, and the extensive comments spliced into the program provide further documentation.

VARIABLES

A: Used to position VARPTR of AS in screen storage routine (lines 700-710).

AS: Used to take data from screen into array P\$. (See also A).

AX\$: Current function (S = Set, R = Reset, M = Move, T = Type).

AY\$: Repeat (Y = Yes, N = No).

CS: Code to be printed to printer to cause either condensed or normal printing.

CU: Current cursor position.

CU\$: Current cursor character.

E: Character present at cursor position.

I, J: Miscellaneous loops.

IM: Image number.

MD: Current mode, based on Y position of current pixel within current cursor position.

N: Maximum number of frames.

P\$(x,y): Image storage array.

Q, QQ, Q\$, QQ\$: Miscellaneous user input.

Clears an amount of memory for strings based on the amount of free memory in the machine. Then calculates the maximum number of images there is room to store (N) and reserves array space for these images (P%). Also defines all variables starting with a letter from D to K as integers to speed up execution time.

```
100 CLS: CLEAR50: CLEAR(MEM%.B): N=FRE(A%)/900: DEFINTD-K: DIMP$(12,N):
```

Initialization of some important variables, including the names of the images in storage.

```
105 FORI=1TON:P$(0,I)="UNUSED":NEXTI
110 AX$="M":AY$="N":CU=415:CU%=CHR$(140):MD=2:E=128
```

The main menu of the program. A selection is accepted, acted upon, then line 122 returns to the menu. Use the down-arrow key to align the menu in the format shown.

```
120 CLS:PRINT@20,"MENU
DRAW MODE      1
DISK SAVE      2
DISK READ      3
PRINTER OUTPUT 4
IMAGE REVIEW   5
":INPUT"SELECTION";Q:CLS:ONQGOSUB490,300,350,5800,900
122 GOTO120
```

This is a program development "Can't mess up my file name between saves" programming hint. Just type GOTO199 to save the program (disk users only). When employing this technique, be certain that the program doesn't stumble across the line by mistake (note line 122 which prevents this from occurring.)

```
199 SAVE "GRAPHIC/BAS":STOP
```

Figure out what graphic character needs to be "set" on the video.

```
200 IFE<129THENE=ASC(CU%):RETURNELSEY=ASC(CU%):IFY=(Y+E)+(E=191)THEN
ENRETURN
201 IF((Y=131)*(E=140))+((Y=140)*(E=131))THENE=143:RETURN
202 IF((Y=131)*(E=176))+((Y=176)*(E=131))THENE=179:RETURN
203 IF((Y=131)*(E=188))+((Y=140)*(E=179))+((Y=176)*(E=143))THENE
=191:RETURN
204 IF((Y=140)*(E=176))+((Y=176)*(E=140))THENE=188:RETURN
205 RETURN
```

Logic for the reset option in the draw mode.

```
210 IFE<129THENE=128:RETURNELSEY=ASC(CU%):IFY=ETHENE=128:RETURN
211 IF(Y=131)*(E=140)+(E=176)+(E=188)THENRETURN
212 IF(Y=140)*(E=131)+(E=176)+(E=179)THENRETURN
213 IF(Y=176)*(E=131)+(E=140)+(E=143)THENRETURN
214 IFY=131THENE=E-3ELSEIFY=140THENE=E-12ELSEIFY=176THENE=E-48
215 RETURN
```

Start of save-data routine.

```
300 PRINT@832,;:INPUT "DISK OR TAPE";Q:Q%=LEFT$(Q$,1):IF Q$="D"
THENPRINT@896,"WHAT IS THE FILE NAME TO BE STORED";:ELSEPRINT@89
6,"HIT ENTER WHEN TAPE IS READY TO SAVE";
```

When working with disk files, proper procedure says to close all files before opening any new ones. Then "D" specifies output and the 1 specifies buffer 1. QQ\$ is the name of the disk file.

```
310 INPUTQQ$:IF Q$="D" THEN CLOSE:OPEN"Q",1,QQ$
```

Store the data. If saving to tape, the string "END OF DATA" is printed at the end of the tape so the "read" routine will know when it is done.

```
320 FORIM=1TON:FORI=0TO12
325 IFF$(0,IM)="UNUSED"THENI=12:NEXTI,IM:IFQ$="D"THENCLOSE:RETUR
NELSEPRINT#-1,CHR$(34),"END OF DATA";CHR$(34):RETURN
330 IF Q$="D" THEN PRINT#1,CHR$(34);P$(I,IM);CHR$(34):NEXTI,IM:CL
LOSE:RETURN
332 PRINT#-1,CHR$(34);P$(I,IM);CHR$(34):NEXTI,IM:PRINT#-1,CHR$(3
4);"END OF DATA";CHR$(34):RETURN
```

Beginning of read-data routine.

```
350 PRINT@0,CHR$(31);:INPUT "DISK OR TAPE";Q:Q%=LEFT$(Q$,1):IFQ
$="D"THENPRINT@0,"WHAT IS THE FILE NAME TO BE RECALLED";:ELSEPRI
NT@0,"HIT ENTER WHEN TAPE IS READY TO LOAD";
```

If disk, then open the previously created file for input ("I") using buffer 1.

```
360 INPUT QQ$:PRINT@64,STRING$(64,"-"):IM=1:I=0:IFQ$="D"THENCLOS
E:OPEN"I",1,QQ$
```

Input for disk. EOF(1) is true when the End Of File #1 has been reached, signifying there is no more data to be read.

```
370 IF Q$="D" THEN IFEOF(1)THENRETURN:ELSEINPUT#1,P$(I,IM):GOTO3
74
```

Input for cassette. Each string inputted is compared to "END OF DATA" to check for the end of data stored on the tape.

```
372 INPUT#-1,P$(I,IM):IF P$(I,IM)="END OF DATA" THEN P$(I,IM)="U
NUSED":RETURN
```

Display the inputted data on the screen, then continue inputting.

```
374 PRINT@128+I*64,CHR$(31);P$(I,IM);:I=I+1:IFI=13THENI=0:IM=IM+
1
376 GOTO 370
```

Draw a line of hyphens to separate the graphic design on the upper part of the screen from the list of options at the bottom, which are printed by the subroutine at line 600.

```
490 PRINT@768,STRING$(64,"-");:GOSUB600
```

This is the point where most input from the keyboard will be



continued on next page

continued from previous page

made. Q\$ stores any new strike on the keyboard. The PEEK(14400) statements are checking if any of the arrow keys are being held down. If they are, they are converted to the proper ASCII codes and put into Q\$ for later interpretation.

```
500 PRINT@CU,CU$;Q$=INKEY$;IF Q$=""ANDAX$(">"T"ANDAY$="Y"THENIF(
PEEK(14400)AND8)THENQ$=CHR$(91)ELSEIF(PEEK(14400)AND16)THENQ$=CH
R$(10)ELSEIF(PEEK(14400)AND32)THENQ$=CHR$(8)ELSEIF(PEEK(14400)AN
D64)THENQ$=CHR$(9)
503 POKE 15360+CU,E;IF Q$="" THEN 500
504 IFAX$="T"THENIFQ$=CHR$(31)THENAX$="M":GOSUB610;ELSEIFQ$(">"CHR
$(10)ANDQ$(">"CHR$(9)ANDQ$(">"CHR$(8)ANDQ$(">"["THENPRINT@CU,Q$;E=AS
C(Q$);Q$=CHR$(9)
```

Checks if the up arrow key was pushed. If so, move the cursor up one graphic row and check for screen roll-over. Note that moving up on the screen is done by decreasing the y coordinate.

```
505 IFQ$="["THENMD=MD-1;GOSUB590
```

Check if the down arrow was depressed. If so, move the cursor down one graphic row and check for screen roll-over. Note that moving down on the screen is done by increasing the y coordinate.



```
507 IFQ$=CHR$(10)THENMD=MD+1;GOSUB590
```

If the left arrow has been pushed, and the cursor is not at the upper-left of the screen (CU=0), move the cursor one space to the left.

```
510 IF(Q$=CHR$(8))*(CU>0)THENCU=CU-1;E=PEEK(15360+CU)
```

Same as above, except for the right arrow and the cursor at the lower-right of the screen (CU=767).

```
515 IF(Q$=CHR$(9))*(CU<767)THENCU=CU+1;E=PEEK(15360+CU)
```

If the Set command was given (Q\$="S") or we are in the automatic set function (AX\$="S") then go to the setting routine.

```
520 IF(Q$="S")+((AX$="S")*((Q$="["+(Q$=CHR$(8))+(Q$=CHR$(9))+(Q
$=CHR$(10))))THENGOSUB200;POKE15360+CU,E
```

Same as above, but for the Reset command (Q\$="R") and reset function (AX\$="R")

```
525 IF(Q$="R")+((AX$="R")*((Q$="["+(Q$=CHR$(8))+(Q$=CHR$(9))+(Q
$=CHR$(10))))THENGOSUB210;POKE15360+CU,E
```

If the Repeat command is given, toggle the repeat flag (AY\$) to either Y or N and display the change on the screen (GOSUB 600).

```
530 IFQ$="G"THENIFAY$="N"THENAY$="Y";GOSUB600;ELSEAY$="N";GOSUB6
00
```

If the Function command is given, input the new function (AX\$) and display the change on the screen (GOSUB 600).

```
535 IFQ$="F"THENPRINT@832,CHR$(31);PRINT@896,"FUNCTION:--> SET
= S, RESET = R, MOVE = M, TYPE = T";INPUTAX$;GOSUB600
```

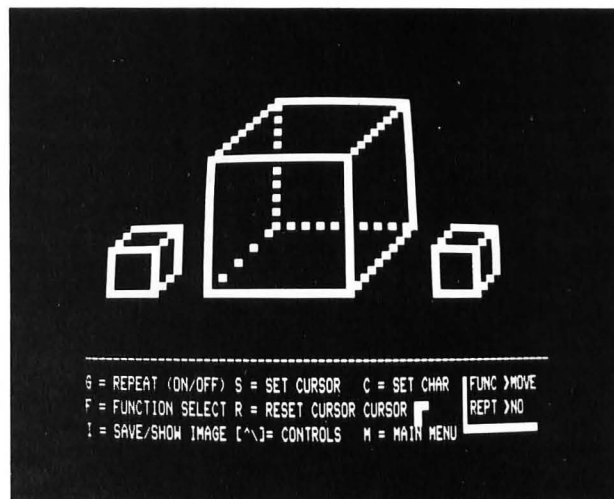
Call the screen I/O subroutine if "I" has been pressed.

```
540 IFQ$="I"THENGOSUB690
```

If a keyboard character is desired rather than a graphic character, allow it to be inputted. This also allows the display of characters not on the keyboard (such as the arrow keys and the underscore).

```
545 IFQ$="C"THENGOSUB650
```

If requested ("M" is typed), return to the main menu.



```
550 IFQ$="M"THENRETURN
```

Return to the option input routine.

```
580 GOTO500
```

Logic to cover situations where the cursor actually crosses screen lines (MD = 0 or 4). The cursor position is adjusted with consideration for screen wrap-around.

```
590 PRINT@CU,CHR$(E);IF(MD=0)*(CU>63)THENMD=3;CU=CU-64;ELSEIF(M
D=4)*(CU<703)THENMD=1;CU=CU+64;ELSEIF(CU<64)*(MD=0)THENCU=704+CU
;MD=3ELSEIF(MD=4)*(CU>704)THENMD=1;CU=CU-704
591 E=PEEK(15360+CU)
```

Define the proper cursor according to the the mode (MD).

```
595 IFMD=1THENCU$=CHR$(131)ELSEIFMD=2THENCU$=CHR$(140)ELSEIFMD=3
THENCU$=CHR$(176)
596 PRINT@941,CU$;RETURN
```

This subroutine sets up the list of possible options at the bottom of the screen.

```
600 PRINT@832,CHR$(31);PRINT@832,"G = REPEAT (ON/OFF)
```



```

F = FUNCTION SELECT
I = SAVE/SHOW IMAGE";PRINT@852,"S = SET CURSOR";PRINT@916,"R =
RESET CURSOR";
602 PRINT@980,"[";CHR$(94);CHR$(92);CHR$(93);"= CONTROLS";PRINT
@933,"CURSOR ";CHR$(191);CU#;
605 PRINT@869,"C = SET CHAR.";PRINT@997,"M = MAIN MENU";
610 PRINT@883,CHR$(149);"FUNC >";IF AX$="T"THENPRINT"TYPE ";EL
SEIFAX$="S"THENPRINT"SET ";ELSEIFAX$="R"THENPRINT"RESET";ELSE
PRINT"MOVE ";
615 PRINT@947,CHR$(149);"REPT >";IF AY$="Y" THEN PRINT"YES";EL
SEPRINT"NO ";
618 PRINT@1011,STRING$(11,131);
620 RETURN

```

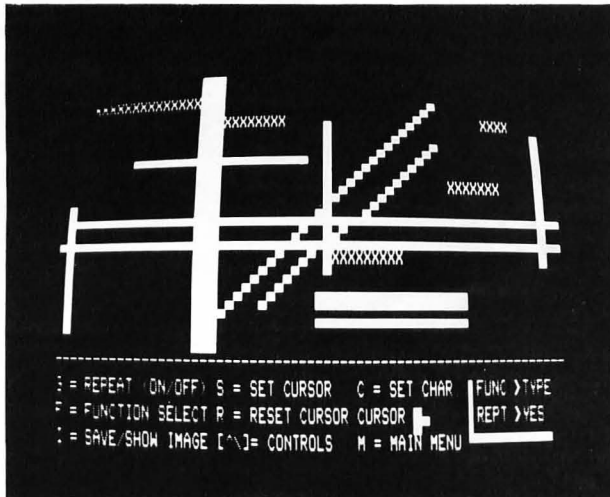
Subroutine for single character input. Either the keyboard character, or the ASCII code of a character, may be inputted and displayed.

```

650 PRINT@832,CHR$(31);PRINT@896,"CHARACTER";INPUTQ$;IF LEN(Q$
)>1 THEN IF VAL(Q$)<32 THEN Q$=LEFT$(Q$,1); ELSE Q=VAL(Q$);IF Q<
=128 OR Q=131 OR Q=140 OR Q=143 OR Q=176 OR Q=188 OR Q=191 OR Q=
179 THEN Q$=CHR$(Q);ELSE 650
652 GOSUB600;PRINT@CU,Q$;E=PEEK(15360+CU);RETURN

```

This routine searches the stored image array for the first available location.



```
680 FORIM=1TON:IFP$(1,IM)=""THENRETURNELSENEXTIM:RETURN
```

Informs the user where the image will be stored, and accepts a name for the image to be saved. If there is no room for additional storage, the user will be so informed.

```

690 GOSUB680;PRINT@832,CHR$(31);IFIM<NTHENPRINT@832,"THE NEXT
AVAILABLE IMAGE # IS";IM:INPUT"WHAT IS THE IMAGE TITLE (8 CHRS)"
;Q$;Q$=LEFT$(Q$,8);ELSEINPUT"THERE IS NO UNUSED IMAGE SPACE. HI
T ENTER";Q:GOSUB600;RETURN

```

Check all the other image labels and see if there is already an image by the same name. If so, the user is alerted and given the opportunity to either change the name or authorize the new image to replace the old.

```

695 FOR J=1TON:IFP$(0,J)=Q$THENPRINT@896,"NAME ALREADY IN USE, <
R> REPLACE, <N> NEW IMAGE NAME";INPUTQ0$;IFQ0$="N"THENGOTO690EL
SEIM=JELSENEXTJ
697 P$(0,IM)=Q$

```

Scan video RAM and build the screen display into strings which

are stored in the array P\$. The program uses the VARIABLE Pointer function to tell BASIC that the string A\$ resides in video memory where the screen display is stored.

```

700 A$="";FORI=0TO11:PRINT@999,"LINE";I+1;
705 A=VARPTR(A$);POKEA,64;POKEA+1,(I-INT(I/4)*4)*64;POKEA+2,60+I
NT(I/4)
710 P$(I+1,IM)=A$;NEXTI;GOSUB600;RETURN

```

Gives an index of all images in storage, and allows the user to delete or modify requested images.

```

900 CLS;MK$="## % % ";PRINT@26,"IMAGE INDEX"
902 FOR I=1TON:PRINTUSINGMK$;I,P$(0,I);NEXTI
904 PRINT@832,;:INPUT"WHICH IMAGE # DO YOU WISH TO VIEW";IM
905 GOTO 2000
906 PRINT@832,CHR$(31);"DO YOU WANT TO VIEW ANOTHER IMAGE (Y/N)"
;:INPUT Q$;IF LEFT$(Q$,1)="Y" THEN 900
910 RETURN

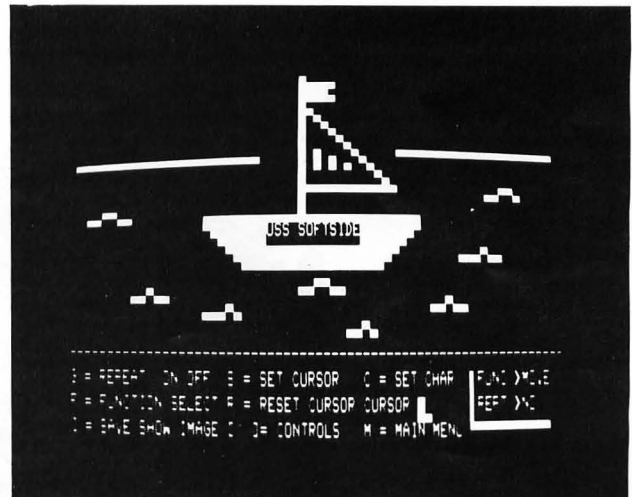
```

Display the chosen image and input what the user desires to do with it.

```

2000 Q$="";PRINT@0,;:FORI=1TO12:PRINT$(I,IM);NEXTI:PRINTCHR$(3
1);PRINT@896,"PRESS <E> TO ERASE, <M> TO MODIFY, <ENTER> TO CON
TINUE";INPUTQ$;IFQ$=""THEN 906

```



```
2005 IF Q$="M" THEN 490
```

```
2010 P$(0,IM)="UNUSED";FORI=1TO12:P$(I,IM)=""NEXTI;GOTO 906
```

Print an image to a printer. To do this requires a printer with the ability to print graphics, such as the Microline-80 or the MX-80. C# is used to hold the codes needed to cause the printer to print in either normal or condensed print mode. If you are using an MX-80, be sure to have it switched to S-80 mode before printing.

```
5800 PRINT:PRINT"CONDENSED REPORT - 1, NORMAL REPORT - 2, MENU -
3";INPUTQ1:IF Q1<>1 AND Q1<>2 THEN RETURN
```

```
5802 C$=""
```

```
5805 PRINT "ENTER TYPE OF PRINTER YOU HAVE:
```

```

MICROLINE 80 1
MX-80 2
OTHER 3

```

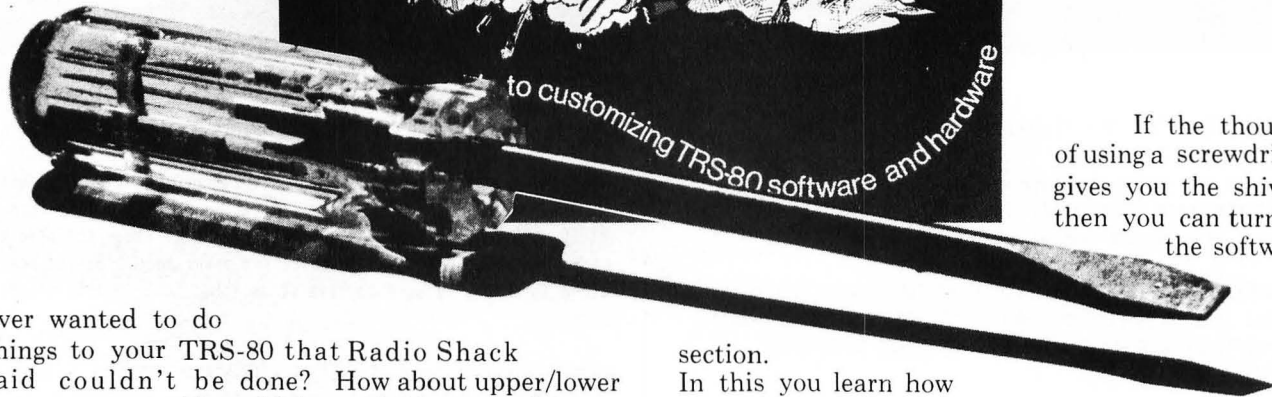
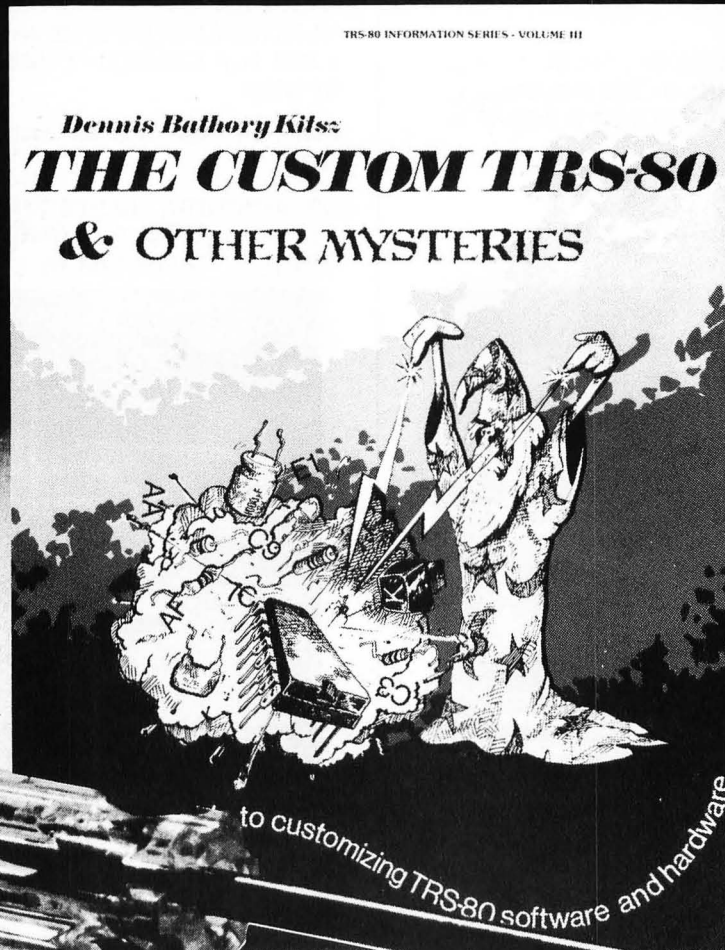
```
SELECTION";INPUT Q
```

```
5810 IFQ1=1THENIFQ=1THENC$=CHR$(29);ELSEIFQ=2THENC$=CHR$(15);ELS
EIFQ=1THENC$=CHR$(30)ELSEIFQ=2THENC$=CHR$(18)
```

```
5850 PRINT:INPUT"WHAT IMAGE WOULD YOU LIKE TO PRINT";X
```

```
6000 LPRINTC$;FORI=0TO12:LPRINT$(I,X);NEXTI;RETURN
```

TUNE-UP YOUR TRS-80



to customizing TRS-80 software and hardware

If the thought of using a screwdriver gives you the shivers then you can turn to the software

Ever wanted to do things to your TRS-80 that Radio Shack said couldn't be done? How about upper/lower case, reverse video, high-resolution graphics, a high-speed clock, audible keystrokes, an extra keyboard, and a real-time clock? Still not enough? How about using an 8-track as a mass storage device, making music, controlling a synthesiser, monitoring your data bus, and individual reverse characters? All these hardware modifications, plus lots more, are in **The Custom TRS-80 and Other Mysteries**, by **Dennis Bathory Kitz** - the latest book from **IJG Computer Services**.

section. In this you learn how to make BASIC programs auto-execute, reset the memory size, patch into the interpreter, test memory with machine-language, pack program lines with machine code, and generate sound effects.

The Custom TRS-80 and Other Mysteries is more than 300 pages of practical information, and tested software, for **\$29.95**. Order your copy now, and start turning your TRS-80 into a five-hundred-dollar supercomputer! #65-238001B \$29.95



TSE-HARDSIDE

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

SoftSide August 1981



APL

by Phelps Gates

Now a high-level, scientific programming language for the home computer that doesn't cost \$200 or \$300. The power of this language is in its strong mathematical operations, especially with regard to matrices and vectors. Programs requiring matrix multiplication or other matrix problem solving that would require hours of programming time in BASIC are solved quickly and with minimal effort in APL.

To aid in learning APL, lessons are included on the disk. Starting from the basics, you are brought step by step through the various programming techniques involved with APL. These lessons act as a tutor which will have you "talking APL" in no time. Also available is the book, "APL: An Interactive Approach," which reinforces many of the examples given in the lessons and provides additional insight into APL programming.

FEATURES

APL-80 on disk contains the following features:)SAVE and)LOAD workspace on disk;)COPY other workspaces into current ones; Return to DOS for directory or commands without losing your workspace; Send output to lineprinter; Five workspaces of lessons included; Sequential and random files; 15 digit precision; Monadic and dyadic transposition; Easy editing within FUNCTION lines; Latent expressions (FUNCTION can "come up running" when loaded); Tracing of function execution; Real-time clock; User-control of random link; Workspace is 25587 bytes (in 48K machine); Arrays may have up to 63 dimensions.

COMMANDS APL-80

APL-80 supports the following commands; Absolute value, add, and assign, branch, catenate, ceiling, chr\$/asc, circular, combinational, comment, compress, deal, decode, divide, drop, encode, equal, expand, exponential, factorial, floor, format, grade down, grade up, greater, greater/equal, index generator, indexing, index of, inter product, label, less, less/equal, logarithm, maximum, member, minimum, multiple, nand, negate, nor, not, not equal, or, outer product, peek, poke, quad, quote quad, random, ravel, reciprocal, reduction, reshape, residue, reverse, rotate, scan, shape, sign, system, subtract, take, transposition.

SPECIFICATIONS

Minimum system requirements: 32K disk system (&48K recommended) includes APL-80, Five workshapes of lessons, instruction manual. \$39.95 on disk

Reduced feature: 16K Level II tape version, no lessons.

Transpositions, format, and inner product not implemented. Reduced domain for some functions, 6 digit accuracy.

..... \$14.95 on cassette

LIMITATIONS

Due to the absence of the special APL character set on the TRS-80 , APL-80 uses shifted letters to represent the various APL characters. In addition to the keyboard limitations, lamination, domlno, and matrix inverse are not implemented but can be derived with user-defined functions. Multiple specifications must be split into two statements unless the left-hand assignment is to a quad. This also applies to implied multiple specifications. Reduction and reshape (p) are not permitted for empty arguments; the argument of add/drop may not be scalar; empty indices are not permitted. A quad (q) can't be typed in response to a quad (nor can the name of a function which itself gets input from a quad). Quote-quad (m) is permitted. No more than 32 user functions can be defined in a single workspace and a function may not contain more than 255 lines.

A comment (c) must occupy a separate line: a comment can't follow a function statement on the same line. In the tape version, arrays are limited to five (5) dimensions.

**SoftSide
Selections**
6 South Street Milford NH 03055
For Orders Only 803-673-0585



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

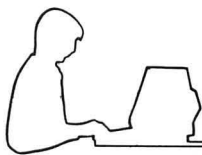
THURS-SUN
OCT 29-NOV 1, 1981

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



Shark

by Mark Cross

“Shark” is a Hi-Res graphics game requiring Applesoft and 16K RAM.

If you need a break from shooting down assorted space vessels and invading aliens, this is just the program for you. It’s a game of pursuit in which (for a change) you are the pursuer. With your game paddle or keyboard you control a 2000 pound aquatic eating machine which just loves all the little fish that are swimming around on your video screen.

The most satisfying thing about “Shark” is that the other fish can’t do a thing to get back at you. For once, you’re the bully on the block, and you can wipe out all the little guys with no threat of revenge! You do have a time limit, and at the higher difficulty levels it’s quite a challenge to finish your supper before the end of the game.

At the beginning of the game you choose not only your skill level 0 through 9, but also whether you want to use Paddle 0 or the keyboard to maneuver the shark. The two types of control require different types of skill, and both are enjoyable. If you use the paddle, turning the knob will

move you upward or downward as you swim from side to side, and pressing the button will reverse your direction. The keyboard, on the other hand, controls movement through the “U” and “D” keys for up-and-down movement, and the “T” for turning around. Pressing “U” or “D” repeatedly will change your angle of attack, so that you can achieve quite accurate control with some practice.

Does this game have any socially redeeming value? Probably not. But it’s sure a lot of fun.

VARIABLES

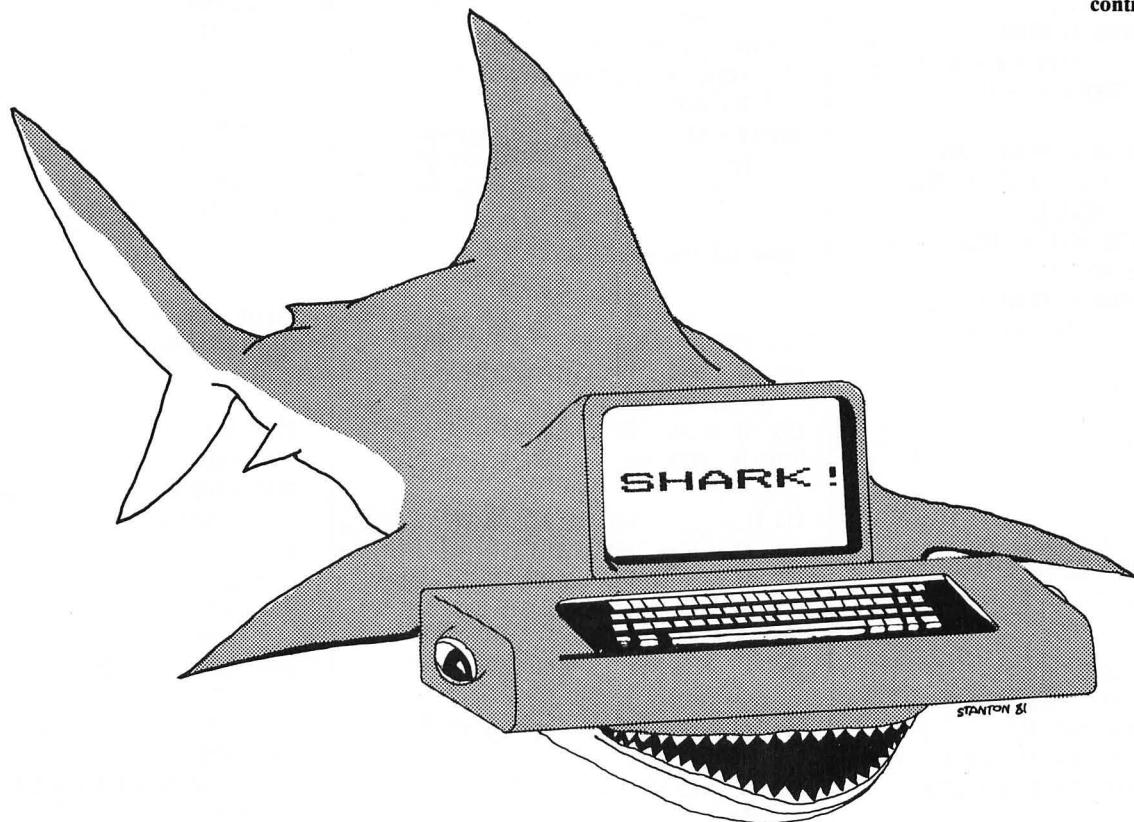
ZERO,WUN,TWO,THREE,FOUR,FIVE,SIX: Variables used in place of the corresponding constants.
BOT: Bottom limit of fish motion (= 157).
DD: Distance at which small fish sees shark and begins to flee.
DOWN: ASCII value of “D”.
I,II,J: Loop counters.
KE: Keyboard buffer address (= -16384).
KFLAG: Control flag; = 1 if keyboard control, = 0 if paddle control.

KK\$: Keyboard input character.
NB: Number of small fish left.
RJ: Right screen limit.
SB: Keyboard-clear address (= -16368).
SKILL: Skill level.
SW: Address of switch on Paddle 0 (= -16287).
T: Time.
T1,T2: Set to 21 and 35; used in place of constants.
TFLAG: Turn flag: direction of shark’s movement, + 1 or -1.
TURN: ASCII value of “T”.
UP: ASCII value of “U”.
VB: Maximum speed of the small fish.
VZ: Maximum speed of the shark.
V,W,X,Y: Speed and position of the shark.
VB(*),WB(*),XB(*),YB(*): Speeds and positions of all surviving small fish.
VV,WW,XX,YY: Temporary speed and position of small fish.

SHAPES

1: Shark facing right.
2: Shark facing left.
3: Small fish facing right.
4: Small fish facing left.

continued on next page



continued from previous page

10 GOTO 2000

Subroutine to generate eating buzz.

```
100 FOR J = 1 TO 40:JJ = PEEK (
    - 16336): NEXT J: RETURN
```

Subroutine to print number of fish left.

```
110 VTAB 22: HTAB SIX: PRINT " "
    ; INT (NB - 1); " "; RETURN
```

Subroutine to turn shark around.

```
120 VTAB 24: HTAB WUN: PRINT "TU
    RN "; CHR$(7);:TFLAG = - SGN
    (V) - (V = ZERO)
130 VTAB 24: HTAB WUN: PRINT "
    ";: RETURN
```

Keyboard up-down routine.

```
200 VTAB 24: HTAB 1: PRINT " UP
    ";
210 FOR J = 1 TO 10:JJ = PEEK (
    - 16336): NEXT J
220 VTAB 24: HTAB 1: PRINT "
    ";:W = W - 3: IF
    W < - VZ THEN W = - VZ
230 RETURN
250 VTAB 24: HTAB 1: PRINT " DOW
    N";: FOR J = 1 TO 10:JJ = PEEK
    (- 16336): NEXT J
260 FOR J = 1 TO 10:JJ = PEEK (
    - 16336): NEXT J
270 VTAB 24: HTAB 1: PRINT "
    ";:W = W + 3: IF ABS (W) >
    VZ THEN W = VZ
280 RETURN
```

Move shark.

```
400 T = T - WUN: VTAB T1: HTAB TT
    : PRINT T;" ";: IF T < WUN THEN
    1000
410 IF KFLAG = ZERO THEN 480
420 KK = PEEK (KE): POKE SB,ZERO
    : IF KK < MID THEN 500
430 IF KK = TURN THEN IF ABS (
    V) > TWO THEN GOSUB 120: GOTO
    500
```

```
440 IF KK < > UP THEN 460
450 GOSUB 200: GOTO 500
460 IF KK = DOWN THEN GOSUB 250
```

```
470 GOTO 500
480 IF PEEK (SW) > = MID THEN
    GOSUB 120
490 W = VZ * ( PDL (ZERO) - MID) /
    MID
500 XDRAW WUN + (V < ZERO) AT X,
    Y
510 V = TFLAG * VZ
520 X = X + V: IF X < FOUR THEN X
    = FOUR
530 IF X > RJ THEN X = RJ
540 Y = Y + W: IF Y < FIVE THEN Y
    = FIVE
550 IF Y > BOT THEN Y = BOT
560 XDRAW WUN + (V < ZERO) AT X,
    Y
```

Check for caught fish.

```
570 FOR I = WUN TO NB
580 IF ABS (YB(I) - Y) > THREE THEN
    600
590 IF ABS (XB(I) - X) < SIX THEN
    II = I: I = 999
600 NEXT I: IF I < RJ THEN 670
610 GOSUB 100: GOSUB 110
620 XDRAW THREE + (VB(II) < ZERO
    ) AT XB(II),YB(II)
630 IF II = NB THEN 660
640 FOR J = II TO NB - WUN:Z9 =
    J + WUN
650 XB(J) = XB(Z9):YB(J) = YB(Z9)
    :VB(J) = VB(Z9):WB(J) = WB(Z
    9): NEXT J
660 NB = NB - 1: IF NB < 1 THEN 1
    000
```

Move all the small fish.

```
670 FOR I = WUN TO NB
680 XX = XB(I):YY = YB(I):VV = VB
    (I):WW = WB(I)
690 IF KFLAG = ZERO THEN 740
700 IF PEEK (KE) < MID THEN 750
710 KK = PEEK (KE): POKE SB,0: IF
    KK = DOWN THEN GOSUB 250
720 IF KK = UP THEN GOSUB 200
730 IF KK = TURN THEN GOSUB 120
735 GOTO 750
740 IF PEEK (SW) > = MID THEN
    GOSUB 120
750 IF ABS (YY - Y) > DD THEN 7
    90
760 IF ABS (XX - X) > DD THEN 7
    90
```

```
770 VV = VB * SGN (XX - X)
780 WW = VB * ( SGN (YY - Y) + (Y
    Y = Y)) * RND (WUN) / PP
790 IF XX > = RJ THEN VV = - T
    WO - VB * RND (WUN): GOTO 8
    10
800 IF XX < = FIVE THEN VV = TW
    O * VB * RND (WUN): GOTO 83
    0
810 IF YY > = BOT THEN WW = -
    TWO - VB * RND (WUN): GOTO
    830
820 IF YY < = FIVE THEN WW = TW
    O + VB * RND (WUN)
830 XDRAW THREE + (VB(I) < ZERO)
    AT XX,YY
840 XO = XX + VV:YO = YY + WW
850 IF XO < FOUR THEN XO = FOUR
860 IF XO > RJ THEN XO = RJ
870 IF YO < TWO THEN YO = TWO
880 IF YO > BOT THEN YO = BOT
890 XDRAW THREE + (VV < ZERO) AT
    XO,YO
900 VB(I) = VV:WB(I) = WW:XB(I) =
    XO:YB(I) = YO
910 NEXT I: GOTO 400
```

End of game; play again?

```
1000 VTAB 23: HTAB 9: PRINT "
    G A M E O V E R": REM
    INSERT CTRL-G AFTER EACH
    LETTER
1010 VTAB 24: PRINT " PLAY AGAIN
    ? ";: GET KK$
1020 IF KK$ = "Y" THEN TEXT : GOTO
    2060
1030 IF KK$ = "N" THEN VTAB 23:
    CALL - 958: END
1040 HTAB 1: GOTO 1010
```

Title page, initialization, instructions.

```
2000 TEXT : HOME : VTAB 7: PRINT
    TAB( 15);"S H A R K"
2010 VTAB 15: PRINT TAB( 24);"B
    Y MARK CROSS"
2020 GOSUB 7000
2030 KE = - 16384:SB = - 16368:
    TURN = 212:UP = 213:DOWN = 1
    96:SW = - 16287
2040 DIM XB(20),YB(20),VB(20),WB
    (20)
2050 FOR I = 1 TO 555: NEXT I
2060 HOME
2070 PRINT "I N S T R U C T I O
    N S"
```



```

2080 PRINT : PRINT "YOU WILL CON
TROL A SHARK WITH"
2090 PRINT "EITHER THE KEYBOARD
OR PADDLE ZERO."
2100 PRINT : PRINT "YOU HAVE TO
CATCH THE SMALL FISH": PRINT
"BEFORE TIME RUNS OUT."
2110 PRINT : PRINT : PRINT "PADD
LE CONTROL:": PRINT "TURN TH
E DIAL TO MOVE UP AND DOWN."

2120 PRINT "PRESS THE BUTTON TO
TURN AROUND. THE BUTT
ON MAKES A BEEP."
2130 PRINT : PRINT "KEYBOARD CON
TROL: U = UP": HTAB 23
: PRINT "D = DOWN": HTAB 2
3: PRINT "T = TURN"
2140 PRINT : PRINT : PRINT "PRES
S P FOR PADDLE CONTROL":
PRINT " OR K FOR KEYB
OARD CONTROL. ";
2150 POKE SB,0: GET KK$: POKE SB
,0:KFLAG = (KK$ = "K")
2160 PRINT CHR$(7);" ";KK$: PRINT

2170 IF KK$ = "K" OR KK$ = "P" THEN
2190
2180 PRINT : PRINT CHR$(7);"ER
ROR": GOTO 2140
2190 PRINT "WHAT SKILL LEVEL, 0-
9? (9 IS HARD.)"
2200 PRINT "PRESS THE NUMBER.
";: POKE SB,0: GET KK$: POKE
SB,0
2210 SKILL = ASC (KK$) - ASC ("
0")

```

```

2220 PRINT SKILL
2230 IF SKILL < 0 OR SKILL > 9 THEN
PRINT "ERROR"; CHR$(7): GOTO
2190

```

Set up all the small fish.

```

3000 NB = INT (5 + (1 + SKILL) *
RND (1));VB = 4 + (1 + SKIL
L) * RND (1)
3010 VZ = 10
3020 IF VB > = VZ - 1 THEN VB =
VZ - 1
3030 HGR : POKE 28,127: CALL 624
54
3040 SCALE= 1: ROT= 0
3050 FOR I = 1 TO NB:XB(I) = 50 +
180 * RND (1):YB(I) = 3 + 1
40 * RND (1)
3060 VB(I) = 10 * RND (1) - 5:WB
(I) = 10 * RND (1) - 5
3070 XDRAW 3 + (VB(I) < 0) AT XB
(I),YB(I)
3080 NEXT I

```

Draw the shark.

```

4000 X = 40 + 200 * RND (1):Y =
20 + 120 * RND (1):V = 20 *
RND (1) - 10:W = 20 * RND
(1) - 10

```

```

4010 TFLAG = SGN (V)
4020 XDRAW 1 + (V < 0) AT X,Y

```

Print the player's score display.

```

5000 HOME : VTAB 22: HTAB 6: PRINT
" ";NB;" FISH LEFT";
5010 VTAB 21: HTAB 1: PRINT "SKI
LL LEVEL = ";SKILL;"
TIME = ";
5020 T = INT (215 - SKILL * 10):
VTAB 21: HTAB 35: PRINT T

```

Initialize the variables which are used in place of constants.

```

6000 DD = 18 + 3 * SKILL
6010 ZERO = 0:WUN = 1:TWO = 2:MID
= 128:THREE = 3:FOUR = 4:FI
VE = 5
6020 TT = 35:LJ = 4:RJ = 275:BOT =
157:SIX = 6
6030 T1 = 21:Z3 = 273:Z4 = 16
6040 PP = (100 - 8 * SKILL) / 60
6050 GOTO 400

```

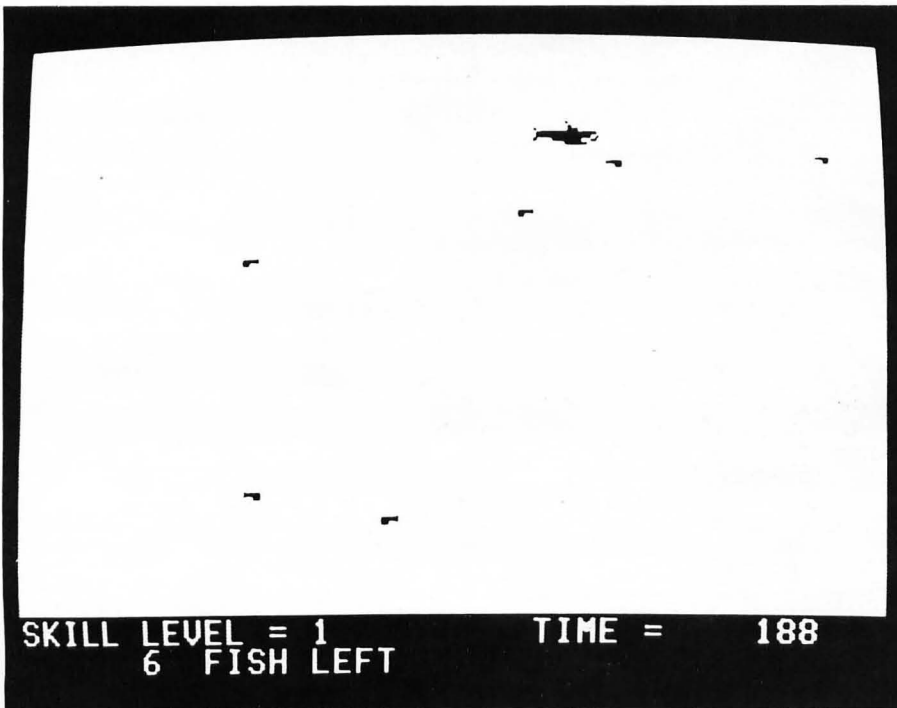
Poke shape table into memory.

```

7000 ST = 768:EN = ST + 125
7010 FOR I = ST TO EN: READ D: POKE
I,D:S = S + D: NEXT
7020 IF S < > 6574 THEN TEXT :
HOME : PRINT "DATA BASE ERR
OR", CHR$(7): END
7030 POKE 232,0: POKE 233,3: RETURN

7040 DATA 4,0,18,0,65,0,112,0,11
9,0,126,0,0,0,0,0,0
7050 DATA 58,63,63,63,60,63,39,6
3,63,55,214,36,12,24,32,172,
50,45,45,45,45,45,36,60,172,
17,46,53,45,45,45
7060 DATA 46,30,63,60,63,63,60,6
2,63,63,77,17,45,45,4,0,59,3
9,103,45
7070 DATA 45,45,44,12,12,12,190,
62,23,45,45,45,45,45,44,3
6,150,18,54,28,32,63,63,63,6
3,63,63,63,63
7080 DATA 119,42,45,45,45,76,9,6
3,63,63,63,39,0,45,62,231,23
1,54,4,0,63,46,101,101,54,4,
0

```



MICRO

The 6502/6809 Journal

- For the serious user of the Apple, PET/CBM, OSI, Atari, AIM, SYM, and KIM.
- The premier 6502 resource journal—now also covering the 6809!
- Internationally respected by professionals in business, industry, and education.



Order today!

We accept VISA & Mastercard.

\$18.00 in the U.S.

\$21.00 foreign
surface mail

(Ask for air rates)

MICRO

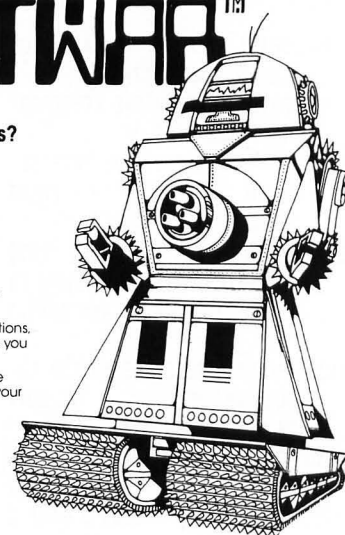
P.O. Box 6502
Chelmsford, MA

01824

(617)256-5515

ROBOTWAR™

- Do you like thinking games?
- Do you like fast-action spectator sports?
- Want to have fun learning more about computers and programming?
- Think you can program better than your friends?



If you answered YES to any of these questions, RobotWar is for you. A game of the future you can play today...

Create a robot by writing a special Battle Language program. This program gives your robot its unique fighting personality.

Debug your robot on the Test Bench, a cybernetic window into your Robot's mind. Is it really checking its damage level to consider evasive action? Does it increment its radar and laser cannon aim while searching for enemies? If all checks out, it's on to...

The Battlefield... Challenge up to four competitors from the Robot Ready Room on your disk. Your robot will meet them in the arena where you have a bird's eye view of the mechanical carnage. Robots scurry about, radars flash, laser shots fly and explode... and only one survives. You're the witness to a futuristic Gladiator spectacle.

Available on disk for the Apple computer with 48K and Applesoft ROM, at computer stores everywhere.

from the leader in quality software

MUSE SOFTWARE™

Apple is a trademark of Apple Computer, Inc. MUSE Software, Inc.

330 N. CHARLES STREET
BALTIMORE, MD 21201
(301) 659-7212

Call or write for information and the name of your nearest MUSE dealer.

This month's special: STARFIGHTER!

The sabre-rattling is over—this is WAR! Your mission: to pilot the SC-78503 STARFIGHTER through the stars in search of the Petro Resource Conglomerate. The PRC disguise their craft as friendly vessels, so you must take care. To help you along a training simulator has been placed at your disposal. It will take hours of training and actual combat to reach the coveted rank of STAR LORD, but as you gain experience with each dogfight, so will you gain the requisite skill. Your flight cassettes or self-booting disk also comes with a top secret manual of 32 pages of induction material for the NEW PILOT. Fight the good fight with STARFIGHTER!

Disk (TRS-80 Mod.1) Reg. \$29.95 SPECIAL - \$22.95

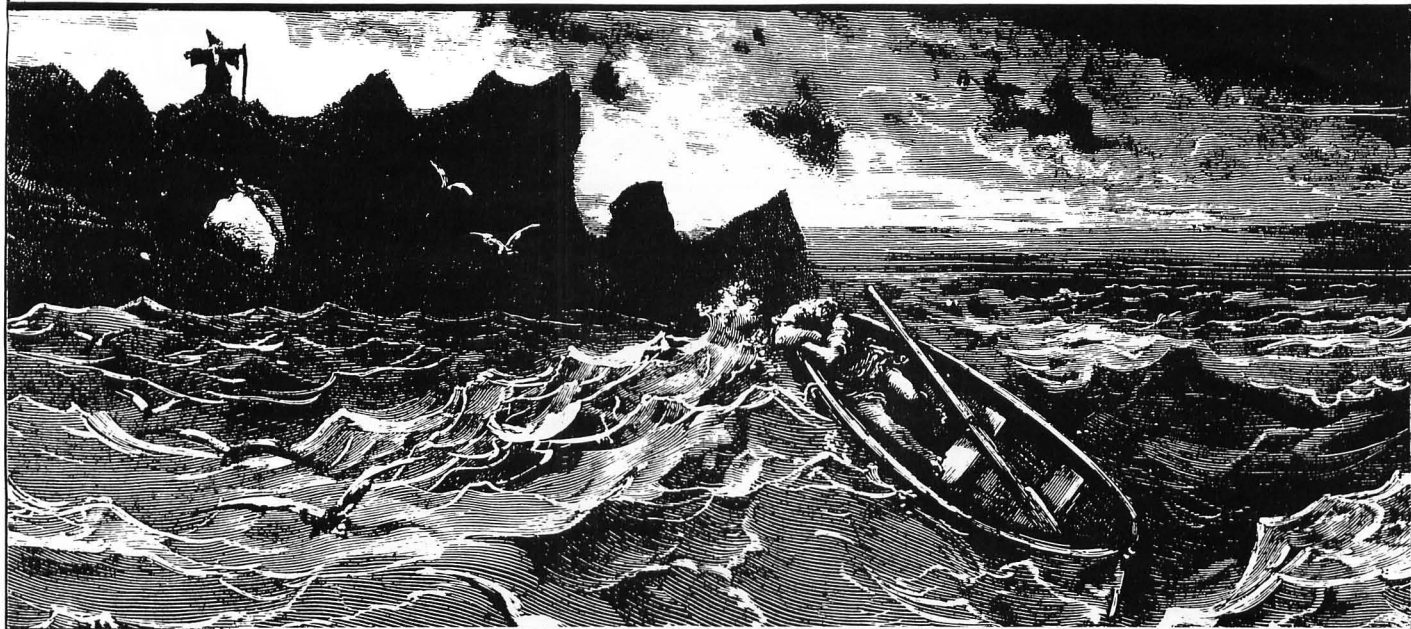
Cassette (TRS-80 Mod.1 & 3) Reg. \$24.95 SPECIAL - \$17.95

OFFER
ENDS
SEPT. 15

SoftSide
Selections

6 South Street, Millis, MA 01955
For Orders Only, 803 671 0565

ADVENTURE OF THE MONTH CLUB



TREASURE ISLAND ADVENTURE

AUGUST ADVENTURE OF THE MONTH

You are a hardy adventurer in search of fame, fortune, and whatever else you can lay your hands on. You find yourself on an island, and you have heard that there is a pirate's treasure here. . . but watch out for the evil magician and the underground torture chamber! You may end up in a spot where all roads leading into it are paved with good intentions. . .

What is Adventure of the Month?

Everybody likes Adventures — they're challenging and entertaining every time you play. But too often, preprogrammed cassettes and disks cost upwards of \$35, a price the manufacturer must charge to defray promotional and packaging costs.

On the other hand, you can enter Adventures yourself, but when you do, you type away all the surprises. As a result, the game loses some of its challenge.

At **SoftSide**, we've found a way to beat the high cost of Adventuring without having to miss out on any of the fun. We're offering no-frill Adventures — high quality Adventures — on cassette or disk at an almost unheard-of price: **\$5 for cassette, \$8 on disk.**

We save you money by only advertising this offer to **SoftSide** readers (you won't see us anywhere else) and by foregoing fancy packaging and documentation — you'll get the software and only the software, but we believe it's as good as the \$50 packages.

You'll save even more by joining the Adventure of the Month Club.

Here's how it works: **SoftSide's** editorial department will select an original BASIC language Adventure each month and make it available to you on a subscription basis:

6 months on cassette: just \$27

6 months on disk: just \$45

Every month we'll tell you about the Adventure you'll be getting in **SoftSide** Magazine. To order, use the convenient order form in this issue — fill it out and send, with payment to:

**Adventure of the Month Club
Department 681
6 South Street
Milford, NH 03055**

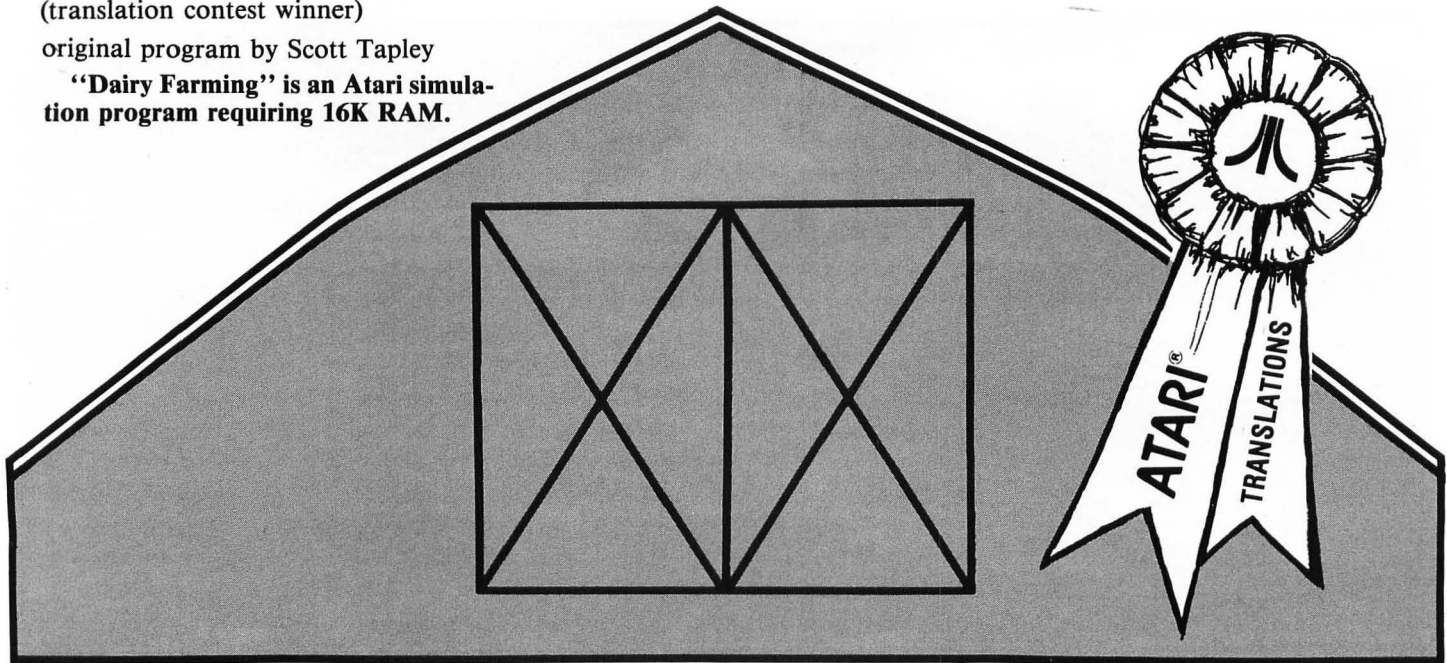


Dairy Farming

by David H. Simmons
(translation contest winner)

original program by Scott Tapley

"Dairy Farming" is an Atari simulation program requiring 16K RAM.



This simulation first appeared as an S-80 program three months ago, in the May issue of **SoftSide**. Now Atari users, too, can experience the challenges of succeeding in agri-business.

You begin your venture with half a million dollars, with which you must buy a farm, cows, and related equipment and supplies. To succeed you must accumulate 10,000 points, based on your total assets of land, cows, money, etc. You begin by buying one of three farms offered to you, and then proceed to the three-phase daily cycle of the simulation.

PHASE ONE

You may undertake any of six activities in the first phase of the cycle: (1) Milk the cows, (2) Buy feed, (3) Buy cows, (4) Feed the cows, (5) Take the cows to pasture, and (6) Sell milk. Feeding the cows is a priority, either in the pasture or with feed you've bought and stored in your silo. Grazing costs you nothing, but it does deplete the grass and you'll need to let the pasture recover for a few days from time

to time. Not feeding the cows will produce two results: First, the cows will produce inferior milk which will have to be thrown out; and second, one of the cows will die later in the day.

When you milk the cows each day, you'll notice that each type gives a different amount of milk (related to the price they command at auction). If your coolers aren't adequate to hold all the milk, some of it will have to be sold quickly (and cheaply). If you have at least 200 gallons of milk, you can sell some or all of it to the milkman, and use the money to accumulate more points.

PHASE TWO

The second phase of each daily cycle involves buying and selling 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.

PHASE THREE

In the third phase of the day's cycle, 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 finally 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.

VARIABLES

A: Acres of land owned.
 AY: Ayrshire cows owned.
 BB: Your bid.
 BID: The current bid.
 BN: Number of barns owned.
 BR: Random increase of bid.
 BS: Brown Swiss cows owned.
 C: Total number of cows owned.
 CE: Indicates whether cows have been fed.
 CLS: Line number of subroutine to clear screen and print heading.
 CM: Total capacity of your coolers, in gallons.
 COW\$: Cow name.
 DA: Day number.
 DEL1,DEL2: Line numbers of delay subroutines.
 ES: Total point value of your silos.
 F: The farm you picked (1-3).
 FV: Current value of your farm.
 G: Gallons of milk in your cooler.
 GU: Guernsey cows owned.
 HO: Holstein cows owned.
 I: Loop counter.
 JE: Jersey cows owned.
 K: Gallons of milk that didn't fit into your cooler.
 L: Miscellaneous loop variable.
 LA: Large coolers owned.
 LIN\$: Graphics string to print line.
 LO: Loan balance.
 MC: Cash on hand.
 MI: Gallons of milk produced that day.
 NUM: Number of cows you're bidding on.
 OP,OP\$: Input from keyboard.
 P: Price per acre of land.
 PB: Used to determine if bid is high enough to buy cow.
 PE: People at the auction.
 PER: Percentage of silo capacity actually filled.
 PV: Number of days of good grass left in your pasture.
 R: Random number.
 S: Cubic feet of feed in your silos.
 SH\$: Color of man's shirt.
 TAB\$: ATASCII code for 1 tab.
 VA: Gallons of milk each cow gave that day.
 VB: Value of farm at beginning of game.
 W: Number of acres bought.
 WAIT: Line number of "Press RETURN to continue" subroutine.
 WP: Your total points.
 WW: Used to value W at end of game.
 XL: Extra land.

```

10 REM DAIRY FARMING
20 REM By Scott Tapley
30 REM Atari conversion: David Simmons
  
```

Initialization of variables.

```

100 CLR :BN=1:DA=1:CM=500:SC=8000:LA=1
:ES=6:WW=900:DEL1=150:CLS=155:WAIT=160
:DEL2=165
110 DIM COW$(11),SH$(7),OP$(28),TAB$(1
),LIN$(39)
115 TAB%=CHR$(127)
119 REM IN LINE 120 SET LIN$ EQUAL TO
      38 CTRL<R>'S PLUS 1 ESC-CTRL<->
120 LIN$="
      "
145 GOTO 200
  
```

Miscellaneous short subroutines.

```

150 FOR I=1 TO 400:NEXT I:RETURN
155 POKE 752,1:PRINT CHR$(125);"DAIRY
FARMING":PRINT LIN$:RETURN
160 PRINT :PRINT "Press [RETURN] to co
ntinue";:INPUT OP$:GOSUB CLS:RETURN
165 FOR I=1 TO 1600:NEXT I:RETURN
  
```

Print introduction and begin with purchase of the farm.

```

200 GRAPHICS 0:GOSUB CLS
205 PRINT "In this simulation of dairy
farming, you will have 500,000 dolla
rs to"
210 PRINT "buy a farm and get started
in your":PRINT "dairying career.":PRIN
T
215 PRINT "The object of the simulatio
n is to":PRINT "accumulate 10,000 poin
ts."
220 PRINT "You get points for such thi
ngs as":PRINT "milk, cash, land, cows,
etc.":PRINT
225 PRINT "The first thing you do is b
uy a farm. Each farm consists of 1 bar
n, 1 small"
230 PRINT "house, 1 large silo, 1 refi
gerated":PRINT "milk cooler, and 1 hay
shed.":PRINT
235 PRINT "The only difference is the
number of acres that the farm has for
grazing."
240 GOSUB WAIT:PRINT "FARM #":TAB$;"AC
RES":TAB$;"PRICE":PRINT TAB$;"1":TAB$;
"140":TAB$;"$439,000"
245 PRINT TAB$;"2":TAB$;"110":TAB$;"$3
99,000":PRINT TAB$;"3":TAB$;"95":TAB$;
"$369,000":PRINT
250 TRAP 250:PRINT "Which farm do you
want to buy (1-3)":INPUT F
255 IF F<1 OR F>3 THEN 250
260 GOTO 260+F
261 A=140:NC=61000:FV=439000:PV=-20:GO
TO 265
  
```

continued on next page

continued from previous page

```
262 A=110:MC=101000:FV=399000:PV=-12:G
DT0 265
263 A=95:MC=131000:FV=369000:PV=-7
265 VB=FV:W=A:PRINT :PRINT "Now it's t
ime to go to the auction":PRINT "and b
uy some cattle...":GOSUB DEL2
270 GOSUB DEL2:GOSUB CLS:GOSUB 1000:GO
SUB CLS:PRINT "Now that you are done b
uying your"
275 PRINT "cattle, you must milk them,
and":PRINT "sell the milk for a profi
t.":PRINT
280 PRINT "Now let's go milk them cows
...":GOSUB DEL2:GOSUB DEL2
```

Display current status and give activities options.

```
300 GOSUB CLS:MC=INT(MC):PRINT "DAY: "
;DA;TAB$;"YOU HAVE $";MC:PRINT "YOUR F
ARMING EQUIPMENT CONSISTS OF:"
305 PRINT "1 no. ";F;" farm, with ";A;
" acres of land":PRINT "and ";BN;" bar
n(s).":PRINT LIN$
310 PRINT "You have ";INT(G);" gallons
of milk":PRINT "in your ";CM;" gallon
cooler(s).
315 PER=S/SC:PRINT "Your silo(s) are "
;(PER*100);"% full":PRINT LIN$
320 C=HD+BS+AY+GU+JE:PRINT "NUMBER OF
COWS:":PRINT "Holstein ";HO;TAB$;"Brow
n Swiss ";BS
325 PRINT "Ayrshire ";AY;TAB$;"Guernse
y ";GU:PRINT "Jersey ";JE;TAB$;"T
OTAL: ";C:PRINT LIN$
330 PRINT "1. Milk cows (Feed 1st!) 2.
Buy feed":PRINT "3. Buy cows at aucti
on 4. Feed cows"
335 PRINT "5. Take cows to pasture 6.
Sell milk":PRINT "Enter option (0 to
continue)";
340 TRAP 340:INPUT OP:IF OP<0 OR OP>6
THEN 340
345 IF OP=0 THEN 400
350 GOSUB CLS:ON OP GOSUB 1200,1300,10
00,1400,1500,1600
355 GOTO 300
```

Routine for other purchases.

```
400 GOSUB CLS:PRINT "AT THIS TIME YOU
MAY BUY ANY THINGS NECESSARY FOR YOU
R FARM."
405 IF C>BN*35 THEN PRINT TAB$;"(You n
eed a bigger barn!)"
410 PRINT LIN$:PRINT "1. Buy more land
for grazing":PRINT "2. Sell land for
quick cash"
415 PRINT "3. Buy a large or small coo
ler":PRINT "4. Buy a bigger barn":PRIN
T "
(Each holds 25 cows)"
420 PRINT "5. Buy a larger silo":PRINT
"6. Sell farm and end game":PRINT "En
ter option number (0 to cont.)";
```

```
425 TRAP 425:INPUT OP:IF OP<0 OR OP>6
THEN 425
430 IF OP=0 THEN 500
435 IF OP=6 THEN 600
440 GOSUB CLS:GOSUB 1900+(OP*100):IF M
C<0 THEN GOSUB 700
445 GOTO 400
```

End-of-day updates; cow check (and killing).

```
500 FV=FV+INT(FV*0.01):WW=WW+10:PA=0:D
A=DA+1:LO=INT(LO+LD*1)
505 IF MC>0 AND LO>0 THEN GOSUB 800
510 IF CE=1 OR C=0 THEN 550
515 R=INT(RND(0)*5):GOSUB 530+R
520 PRINT CHR$(125):POSITION 2,11:POKE
752,1
```



```
525 PRINT "Since you didn't feed your
cows today, one of your ";COW$;" cows d
ied.":GOSUB DEL1:GOSUB DEL1:GOTO 550
530 COW$="Holstein":IF HO>0 THEN HO=HO
-1:RETURN
531 COW$="Brown Swiss":IF BS>0 THEN BS
=BS-1:RETURN
532 COW$="Ayrshire":IF AY>0 THEN AY=AY
-1:RETURN
533 COW$="Guernsey":IF GU>0 THEN GU=GU
-1:RETURN
534 COW$="Jersey":IF JE>0 THEN JE=JE-1
:RETURN
535 POP :GOTO 515
```

Current points; end-of-day rating.

```
550 WP=WP+(HO/3)+(BS/4)+(AY/5)+(GU/6)+
(JE/7)+ES+(A/5)+(LA*2)+SA+(MC/10000)
555 IF LO>0 THEN WP=WP-(LO/100)
560 OP=INT(WP):POKE 752,1:PRINT CHR$(1
25):POSITION 2,11:PRINT "You now have
";OP;" points."
565 PRINT "(",OP/100;"% of your goal)"
:CE=0:IF OP<10000 THEN GOSUB DEL2:GOTO
300
```

```
570 GRAPHICS 18:FOR L=1 TO 10:POSITION
4,2:PRINT #6;" ":POKE 708,5
6+INT(RND(0)*10)*16
575 FOR I=1 TO 60:NEXT I:POSITION 4,2:
PRINT #6;"YOU DID IT!":FOR I=1 TO 75:N
EXT I:SOUND 0,30+RND(0)*50,12,10:NEXT
L
579 REM IN LINES 580-584 TYPE STRINGS
IN INVERSE VIDEO.
580 POKE 710,72:IF DA>125 THEN OP$="NE
EDS IMPROVEMENT-- A LOT!":GOTO 585
581 IF DA>118 THEN OP$="COULD BE BETTE
R.":GOTO 585
582 IF DA>112 THEN OP$="AVERAGE.":GOTO
585
583 IF DA>105 THEN OP$="PRETTY GOOD!":
GOTO 585
584 OP$="A REAL DAIRY FARMER!"
585 PRINT #6;"YOU NOW HAVE OVER":PRINT
#6;"10,000 POINTS!":PRINT #6;"RATING:
":PRINT #6;OP$:POSITION 0,8
590 TRAP 590:POSITION 0,8:PRINT #6;"AN
OTHER GAME (Y/N)?":GET #1,OP:IF OP=89
THEN RUN
593 IF OP=78 THEN END
596 GOTO 590
```

Selling the farm.

```
600 GOSUB CLS:PRINT "When you bought y
our farm it was worth $";VB;" including
the ";W;" acres that"
605 PRINT "came with the farm.":PRINT
"But now it's worth $";INT(FV*1.1):PRI
NT :XL=A-W
610 R=INT(RND(0)*2001):IF R<WW THEN 61
0
615 M=XL*R:IF M>0 THEN PRINT "The land
you bought is worth $";M;":GOTO 625
620 M=0:PRINT "You didn't buy any addi
tional land."
625 R=INT(RND(0)*35):PRINT "Your cows
are worth $";C*(1400+R*35);":PRINT
630 PRINT "You have ";INT(WP);" points
.":PRINT "Are you sure you want to sel
l (Y/N)";
635 TRAP 635:INPUT OP$:IF OP$="N" THEN
RETURN
640 IF OP$<>"Y" THEN 635
645 FV=INT(FV+(FV/9)+MM+C*(1400+R*35))
:PRINT :PRINT "You have just been paid
$";FV;","
650 PRINT "in cash.":PRINT "You now ha
ve $";INT(MC+FV);":PRINT :PRINT "Do
you want to play again (Y/N)";
655 TRAP 655:INPUT OP$:IF OP$="Y" THEN
RUN
660 IF OP$="N" THEN END
665 GOTO 655
Accepting a loan.
700 GOSUB CLS:PRINT "You have spent to
o much money...":PRINT "You will have
to take out a loan."
705 L=1+INT(RND(0)*19):PRINT "You are
$";ABS(MC);" in debt.":PRINT "The curr
ent interest rate is ";L;"%."
```



```

710 PRINT "Pres [RETURN] to accept the
loan";:INPUT OP$
715 LO=LO+ABS(MC):MC=0:L=L/300:RETURN

```

Payment on loan.

```

800 GOSUB CLS:PRINT "LOAN PAYMENT":PR
INT "You have $";MC;","
805 PRINT "And you owe $";LO;" on your
loan.":PRINT "Would you like to make
a payment on your loan (Y/N)";
810 INPUT OP$:IF OP$="N" THEN RETURN
815 PRINT "How much would you like to
pay";:INPUT OP:IF OP<=0 THEN RETURN
820 IF MC<OP THEN PRINT "You don't hav
e enough money!":GOSUB DEL1:GOTO 815
825 LO=INT(LO-OP):MC=INT(MC-OP):IF LO<
=0 THEN PRINT "Your loan is paid off!!
":GOSUB DEL2:RETURN
830 PRINT "You still owe $";LO;" on yo
ur loan.":GOSUB DEL2:RETURN

```

Buying the cows at auction.

```

1000 PRINT "There are 5 different kind
s of cows. (Listed in order of milk p
roduction):"
1005 PRINT "COW #:";TAB$;"TYPE: ";T
AB$;" BID:";PRINT TAB$;"1";TAB$;"Holst
ein";TAB$;"$1500"
1010 PRINT TAB$;"2";TAB$;"Brown Swiss"
;TAB$;" 1400":PRINT TAB$;"3";TAB$;"Ayr
shire";TAB$;" 1325"
1015 PRINT TAB$;"4";TAB$;"Guernsey";TA
B$;" 1250":PRINT TAB$;"5";TAB$;"Jersey
";TAB$;" 1185"
1020 PRINT TAB$;"6";TAB$;"To not bid o
n any cows"
1025 PRINT :PRINT "You have $";MC;". W
hich type of":PRINT "cow do you want t
o bid on (1-6)";
1030 TRAP 1030:INPUT OP:IF OP<1 OR OP>
6 THEN 1030
1031 IF OP=6 THEN RETURN
1035 TRAP 1035:PRINT "How many cows do
you want to buy";:INPUT NUM:IF NUM<1
THEN 1035
1040 GOTO 1040+OP
1041 COW$="Holstein":BID=1500:PB=1500:
GOTO 1050
1042 COW$="Brown Swiss":BID=1400:PB=14
00:GOTO 1050
1043 COW$="Ayrshire":BID=1325:PB=1325:
GOTO 1050
1044 COW$="Guernsey":BID=1250:PB=1250:
GOTO 1050
1045 COW$="Jersey":BID=1185:PB=1185
1050 GOSUB CLS:PE=10+INT(RND(0)*40):PR
INT "There are ";PE;" people here toda
y."
1055 PRINT "The number of people at th
e auction will influence how high yo
u have to"
1060 PRINT "bid to purchase your cow(s
).":PRINT LIN$:PRINT "The bidding on t
he ";COW$

```

```

1065 PRINT "has just started,":PRINT "
with a bid of $";BID;" per cow."
1070 TRAP 1070:POSITION 2,11:PRINT CHR
$(156);CHR$(156);CHR$(156);"The curren
t bid is $";BID;" per cow,"
1075 PRINT "Enter your bid (0 to quit)
";:INPUT BB:IF BB<>0 THEN 1095
1080 R=1+INT(RND(0)*4):GOSUB 1090+R:PR
INT :PRINT "SOLD--":PRINT "to the man
in the ";SH$;" shirt!"
1085 GOTO 1125
1091 SH$="Yellow":RETURN
1092 SH$="Green":RETURN
1093 SH$="Striped":RETURN
1094 SH$="Blue":RETURN
1095 IF BB<BID THEN PRINT "BID TOO LOW
.":GOSUB DEL1:GOTO 1070
1100 IF (BB*NUM)>MC THEN PRINT "YOU DO
N'T HAVE ENOUGH MONEY!":GOSUB DEL1:GOT
O 1070

```



```

1105 IF BB>PB+(PE*25)+RND(0)*500 THEN
GOTO 1115
1110 BID=BB+11+INT(RND(0)*15):GOTO 107
0
1115 PRINT :PRINT TAB$;TAB$;"SOLD!!!":
MC=INT(MC-BB*NUM):GOSUB 1119+OP:GOTO 1
125
1120 HO=HO+NUM:RETURN
1121 BS=BS+NUM:RETURN
1122 AY=AY+NUM:RETURN
1123 GU=GU+NUM:RETURN
1124 JE=JE+NUM:RETURN
1125 PRINT :PRINT "Do you want to buy
any more cows Y/N";
1130 TRAP 1130:INPUT OP$:IF OP$="N" TH
EN RETURN
1135 IF OP$="Y" THEN GOSUB CLS:GOTO 10
00
1140 GOTO 1130

```

Milking the cows.

```

1200 IF PA=1 THEN POSITION 3,12:PRINT
"YOU'VE ALREADY MILKED THEM TODAY!!!":
GOSUB DEL2:RETURN

```

```

1205 PA=1:VA=(2.1+(INT(RND(0)*39)/10))
*2
1210 PRINT "MILK PRODUCTION:"
1215 IF HO>0 THEN PRINT "Your Holstein
cows gave ";INT(HO*VA);" gallons"
1220 VA=VA-0.2:IF BS>0 THEN PRINT "You
r Brown Swiss cows gave ";INT(BS*VA);"
gall."
1225 VA=VA-0.2:IF AY>0 THEN PRINT "You
r Ayrshire cows gave ";INT(AY*VA);" ga
ll."
1230 VA=VA-0.2:IF GU>0 THEN PRINT "You
r Guernsey cows gave ";INT(GU*VA);" ga
ll."
1235 VA=VA-0.2:IF JE>0 THEN PRINT "You
r Jersey cows gave ";INT(JE*VA);" gall
."
1240 MI=(HO*(VA+0.8))+(BS*(VA+0.6))+(A
Y*(VA+0.4))+(GU*(VA+0.2))+(JE*VA):G=6+
MI
1245 PRINT "Total milk production ";MI
;" gallons."
1250 IF CE=0 THEN R=10+INT(RND(0)*28):
PRINT :PRINT "Since you didn't feed yo
ur cows,"
1255 IF CE=0 THEN PRINT R;"% of the mi
lk wasn't good enough":PRINT "and had
to be thrown out.":G=6-MI*(R/100)
1260 IF G>CM THEN PRINT :PRINT "Your c
ooler isn't big enough to hold all of
that milk...You'll have to"
1265 IF G>CM THEN PRINT "sell the extr
a milk at $.75/gallon":K=G-CM:G=CM:MC=
MC+(K*.75)
1270 IF K>0 THEN PRINT :PRINT "You got
";K*.75;" for the extra milk."
1275 K=0:GOSUB DEL2:RETURN

```

Buying feed.

```

1300 PRINT "Your silo is ";PER*100;"%
full.":PRINT "It costs $2.50 a day to
feed a cow."
1305 PRINT "You have ";C;" cows and $"
;MC;".
1310 TRAP 1310:PRINT "How many days wo
rth of food would you like to buy";:IN
PUT OP:IF OP<=0 THEN GOTO WAIT
1315 OP=OP*C*2.5:IF OP>MC THEN PRINT "
You don't have enough money!":GOSUB DE
L1:GOTO 1310
1320 MC=INT(MC-OP):S=S+OP:PER=S/SC:PRI
NT "That cost you $";OP;",";PRINT "You
have $";MC;" left."
1325 IF PER>1 THEN PRINT "Your silo is
full!!!"
1330 GOSUB DEL2:RETURN

```

Feeding the cows.

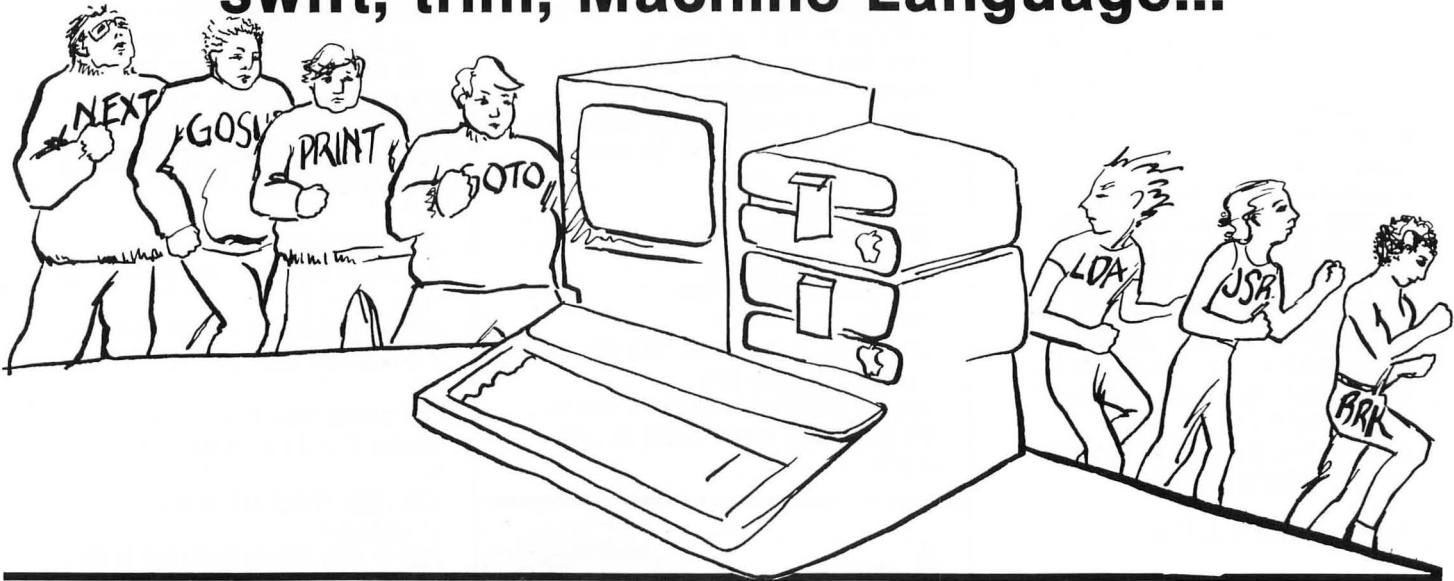
```

1400 IF CE=1 THEN POSITION 4,12:PRINT
"YOU'VE ALREADY FED THEM TODAY!!!":GOS
UB DEL2:RETURN
1405 OP=C*2.5:IF S<OP THEN POSITION 2,
10:PRINT "YOU DON'T HAVE ENOUGH FEED T
O FEED ALLOF YOUR COWS--"

```

continued on page 69

Take the fat off BASIC and turn it into swift, trim, Machine Language...



...without any sweat from you.

INTRODUCING THE HAYDEN APPLESOFT COMPILER For only \$175.00

You can cut the fat from BASIC with the **HAYDEN APPLESOFT COMPILER**, the first true compiler available for the Apple II computer. This multi-pass compiler allows your BASIC programs to run up to ten times faster in Machine Language — faster than your Apple II can normally interpret Applesoft BASIC.

The HAYDEN COMPILER FEATURES:

- * Runs standard Applesoft BASIC programs up to ten times faster.
- * Seventeen-pass compiler produces true machine code.
- * Compiles code at user specified address.
- * Requires only Applesoft in ROM and one disk drive.
- * Handles Hi-Res graphics and shape tables.
- * Arithmetic completed for faster operation.
- * Allows multiple programs to reside in memory at the same time.
- * Works in a single or multiple disk environment or with hard disk.
- * At the end of compilation, the compiler provides a detailed analysis of the structure.

System Requirements

- * A 48K Apple II Plus or Apple II with Applesoft in ROM.
- * The Autostart ROM chip and at least one disk drive.
- * Systems equipped with language system or Microsoft RAMCard.
- * Introductory price for our readers
Save \$25 Limited time only*

\$175.00

The HAYDEN APPLESOFT COMPILER is a product of the Hayden Book Company, Inc.

*offer expires September 15, 1981

SoftSide Selections

6 South Street Milford NH 03055
For Orders Only 603-673-0585



continued from page 67

1410 IF S<OP THEN PRINT "YOU'LL HAVE T
O BUY MORE FEED FIRST.":GOSUB DEL2:RET
URN
1415 PRINT :PRINT :PRINT "Press [RETUR
N] to feed, or D for don't feed";:INPUT
OP\$:IF OP\$="D" THEN RETURN
1420 PRINT "Cows are being fed."
1425 CE=1:PV=PV-1:S=S-OP:PER=S/SC:GOSU
B DEL1:RETURN

Putting cows out to pasture.

1500 IF CE=1 THEN POSITION 4,12:PRINT
"YOU'VE ALREADY FED THEM TODAY!!!":GOS
UB DEL2:RETURN
1505 IF PV>9 THEN PRINT "Your cows hav
e eaten all of the":PRINT "good grass
in the pasture."
1510 IF PV>9 THEN PRINT "In other word
s, you will have to let the grass gro
w for a few days.":GOSUB DEL2:RETURN
1515 IF C>A THEN PRINT "You need to bu
y more land!!":GOSUB DEL2:GOTO 400
1520 PRINT :PRINT "Cows are entering p
asture....":GOSUB DEL1:PRINT :PRINT "C
ows are eating grass....":GOSUB DEL1
1525 PRINT :PRINT "Cows are done eatin
g....":GOSUB DEL1:PRINT :PRINT "Cows a
re going back to barn....":GOSUB DEL2
1530 CE=1:PV=PV+1:RETURN

Selling milk.

1600 IF G<200 THEN POSITION 2,12:PRINT
"YOU DON'T HAVE ENOUGH MILK TO SELL Y
ET":GOSUB DEL2:RETURN
1605 BP=0.9+(INT(RND(0)*36)/10)
1610 PRINT "The local milkman will buy
your milk for \$";BP;" per gallon."
1615 PRINT "You have ";G;" gallons of
milk."
1620 TRAP 1620:PRINT "How many gallons
will you sell";:INPUT OP
1625 IF G<OP THEN PRINT "You don't hav
e that much!!":GOSUB DEL1:GOTO 1620
1630 MC=MC+(OP*BP):G=G-OP:RETURN

Buying land.

2000 P=850+INT(RND(0)*901):PRINT "An a
cre of land will cost you \$";P
2005 PRINT "How many acres do you want
to buy";:INPUT OP:OP=INT(OP):IF OP<1
THEN RETURN
2010 IF MC<OP*P THEN PRINT "You don't
have enough money!!":GOSUB DEL1:GOTO 2
005
2015 MC=MC-OP*P:PRINT "That cost you \$
";OP*P;"":PRINT "you now have \$";MC
2020 A=A+OP:GOSUB DEL2:RETURN

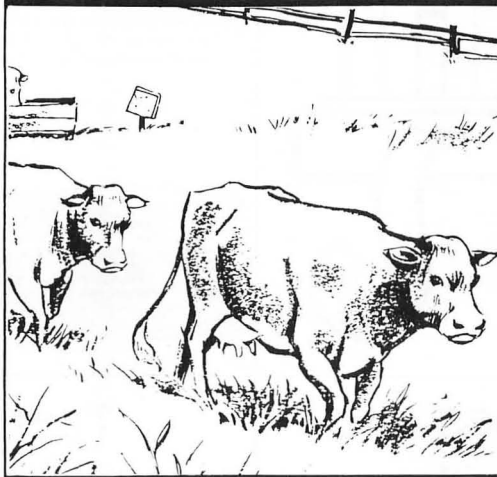
Selling land.

2100 P=950+INT(RND(0)*801):PRINT "You
can sell land for \$";P;" an acre."

2105 PRINT "How many acres do you want
to sell";:INPUT OP:OP=INT(OP):IF OP<1
THEN RETURN
2110 IF OP>A-C THEN PRINT "You need ";
C;" acres for the cows!":GOSUB DEL2:RE
TURN
2115 MC=MC+(OP*P):A=A-OP:PRINT "You no
w have \$";MC:GOSUB DEL2:RETURN

Buying a cooler.

2200 PRINT "Milk Coolers:":PRINT "L =
Large Cooler (500 gallon) \$4,500"
2205 PRINT "S = Small Cooler (200 gall
on) \$2,500":PRINT "N= No Cooler"
2210 PRINT "Which cooler do you want";
:INPUT OP\$:IF OP\$="N" THEN RETURN
2215 IF OP\$="L" THEN MC=MC-4500:CM=CM+
500:LA=LA+1:RETURN



2220 IF OP\$="S" THEN MC=MC-2500:CM=CM+
200:SA=SA+1:RETURN
2225 GOTO 2210

Buying a barn.

2300 PRINT "A barn costs \$5,000.":PRIN
T "(If you don't want it, type: N [RET
])"
2305 PRINT "Press [RETURN] to buy barn
";:INPUT OP\$:IF OP\$<>" THEN RETURN
2310 BN=BN+1:MC=MC-5000:RETURN

Buying a silo.

2400 PRINT "Silo Types:":PRINT "#1. La
rge silo (8000 ft.) \$7,500"
2405 PRINT " 2. Medium silo (7000 ft.)
\$6,000":PRINT " 3. Small silo (6000
ft.) \$5,000":PRINT " 4. No silo"
2410 PRINT "Which silo would you like
to buy";:INPUT OP:IF OP<1 OR OP>4 THEN
2410
2415 GOTO 2415+(OP*5)
2420 MC=MC-7500:SC=SC+8000:ES=ES+6:RET
URN
2425 MC=MC-6000:SC=SC+7000:ES=ES+5:RET
URN
2430 MC=MC-5000:SC=SC+6000:ES=ES+4:RET
URN
2435 RETURN



duplicating service

307 West Main Street
Maple Shade, NJ 08052
(609) 667-1667

- ① ● AMP "Data-settle" blank cassettes for digital use
- ② ● Cassette Storage Boxes
- ③ ● Cassette Labels - Custom printing & blank
- ④ ● Custom Record Album production from your tapes
- ⑤ ● Stereo and Spoken Word cassette duplication

Call or write to:

Jerry

for more information.

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



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
8 1/2 x 11 \$7.95

**SoftSide
Selections**
6 South Street Hilliard NH 03055
For Orders Only 603-873-0585

TRS-80

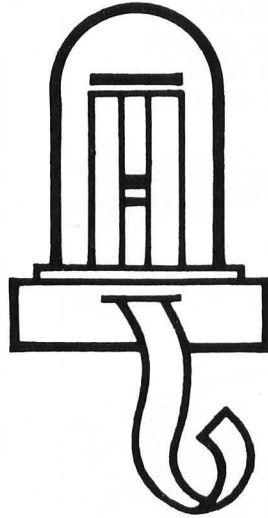
sensational software



Stock & Options Analysis

Cassette CS-3306 (16K), \$99.95
Disk CS-3801 (32K), \$99.95

Should you hedge, buy, or sell out? **Stock and Options Analysis** puts a securities advisor in your computer, providing you with four powerful investment tools. **Option** gives important indices for opening and closing call option transactions. **Opgraph** presents a graph or table of profit for any combination of long or short calls, puts, and stocks. This allows the detailed evaluation of three types of hedges. **Newprem** helps predict the future premiums of an option at any desired time and future stock price. **Portval** lets the computer do the paper work, providing full portfolio services, including value per share, current value, and capital gain. The program includes the effects of commissions, margin interest and dividends. Beyond helping to organize and evaluate your present portfolio, **Stock and Options Analysis** is an excellent aid for planning and testing future strategies. The comprehensive 24-page manual with this package not only shows how the programs work, but is also a primer on the strategy of hedging listed options against common stocks. This strategy has been repeatedly shown to actually be more conservative and more consistently profitable than straight buying and selling of stocks.



Solar Energy Analysis

Cassette CS-3307 (16K), \$49.95
Disk CS-3802 (32K) \$99.95

Available 7/81

F-Chart Solar Energy Analysis eliminates many of the tedious calculations required when designing solar-heating systems. Beyond providing a thermal analysis, the program allows designers to quickly determine the effects of changing any specifications, allowing fast, accurate, and inexpensive experimentation.

Systems using air, liquid, or domestic hot water in any climate can be analyzed in detail. The program expands the traditional F-Chart procedure by taking ground-water temperature into account and allowing for

mixing valves in domestic hot water systems. **F-Chart Solar Energy Analysis** quickly pays for itself by freeing you from time-consuming calculations.

The disk version of the program includes a data base of all necessary climatic data for any location in the United States. These data are in the printed booklet included with the cassette version but must be entered manually for your geographic location.



Personal Address and Information System

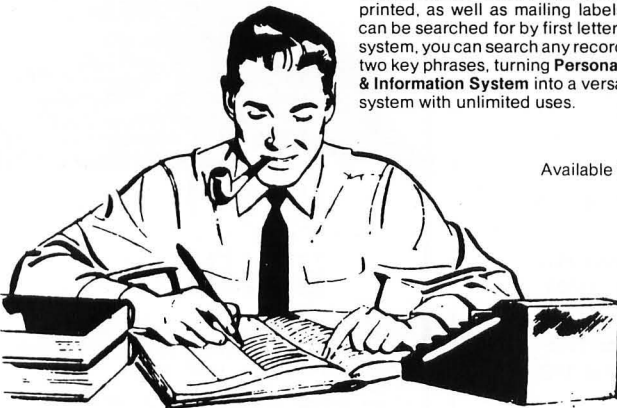
Disk CS-3509 (32K) \$24.95

Is your address book beginning to resemble a heavily-edited inkblot? Do your friends keep moving, forcing you to cross out and rewrite addresses and phone numbers? Let the **Personal Address & Information System**

turn filing drudgery into computing pleasure. You can store all the crucial information, including name, address, home and work phone numbers, spouse's name, and comments or remarks. At any time, the information can be edited or changed.

And there's more. Names can be sorted in alphabetical order. Entire entries can be printed, as well as mailing labels. Names can be searched for by first letter. In a 32K system, you can search any record for up to two key phrases, turning **Personal Address & Information System** into a versatile filing system with unlimited uses.

Available 7/81

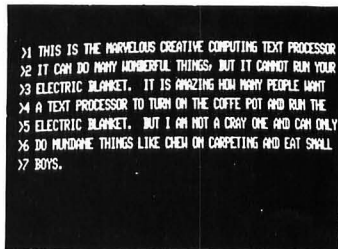


Text Processing

Cassette CS-3302 (16K) \$14.95

CS-3504 Disk (32K) \$24.95
(Disk includes Checking Account, CS-3304)

This program turns a 16K, TRS-80 and lineprinter into a line oriented text-processing system.



COMMANDS	
C	CONTINUE LIST ON SCREEN
D	DELETE LINE
E	EDIT
I	INSERT LINE
K	RESUME KEYING
L	LIST ON SCREEN
P	PRINT HARD COPY
Q	QUIT PROGRAM
T	SAVE ON TAPE
COMMAND?	

Developed exclusively for the TRS-80 this program lets you use the computer to enter general text or business letters, edit and modify your work, save text on cassette tapes, and print out a perfect report, document, or letter every time.

Editing commands are similar to those used in Level II BASIC, so there are no complicated new commands to learn. Lines may be either inserted or deleted. A special format is available to speed entry of business letters. Final printout can be done in numbered pages and you may print multiple copies.

Business Address & Information System

Disk CS-3510 (48K) \$24.95 Available 7/81

Do you need quick access to business contacts and customers? Put more organization in your organization with the **Business Address & Information System**. A complete file containing company name, address, phone number, and comments can be quickly entered and stored. Information can be

changed or edited whenever necessary. The program allows entire entries to be printed, and can also generate mailing labels.

When you need information fast, you can search for specific names or find all entries that contain one or two key phrases. Any key phrases can be used. **Business Address & Information System** will help you make the most of your time, putting the routine work in the computer where it belongs.

Order Today

To order any of these software packages, send payment plus \$2.00 postage and handling per order to Creative Computing, Morris Plains, NJ 07950. Visa, MasterCard and American Express orders may be called in toll-free.

Order today at no risk. If you are not completely satisfied, your money will be promptly and courteously refunded.

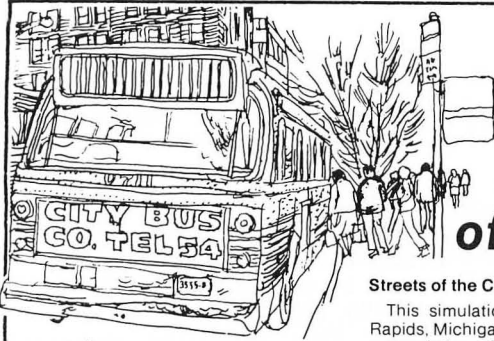
Creative Computing Software
Morris Plains, NJ 07950
Toll-free 800-631-8112
In NJ, 201-540-0445

creative computing software

TRS-80

sensational software

**creative
computing
software**



Trucker and Streets of the City

Streets of the City

This simulation is modeled on Grand Rapids, Michigan, a metropolitan area with a population of 550,000. The budgeting, cost and work standard bases are derived from actual experiences of the city over the past five years. The objective of the simulation is to complete a ten-year plan of street and transit improvements while retaining the support of a majority of the City Commission.

During your tenure, you must construct streets and Interstate highways, repair existing streets, and improve traffic safety. For the Transit Authority you have to upgrade and replace a delapidated bus fleet, increase ridership, reduce maintenance downtime and improve on-schedule performance.

Other factors to be considered are operating tax levies, construction bonding and labor negotiations. The simulation provides a substantial challenge and it is both educational and entertaining.

CS-3207 TRS-80 Cassette (32K) \$24.95
CS-3703 TRS-80 Disk (32K)

Trucker

This program simulates coast-to-coast trips by an independent trucker hauling various cargos. The user may haul oranges, freight or U.S. mail. All have different risks and rewards. Maximum profit comes from prudent risk-taking.

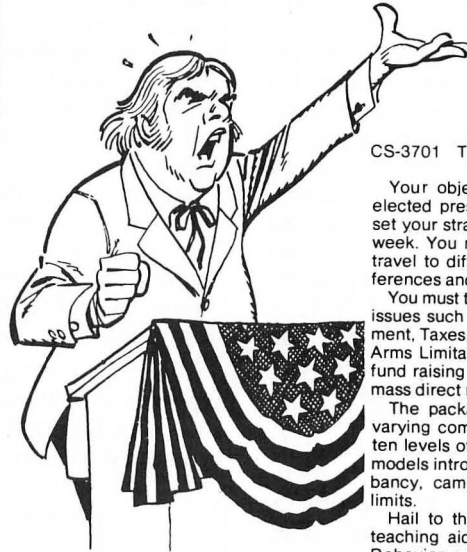
If all goes well, you can obey the speed limits, stop for eight hours of sleep each night and still meet the schedule. Bad weather, road construction or flat tires may put you behind schedule. You may try to increase your profit by skimping on sleep, driving fast or carrying an overweight load.

Other factors are choice of routes, truck payments, fuel, food, tolls and fines. The simulation is engrossing and informative.

Hail to the Chief

by
Phillip W. Brashear
and
Richard G. Vance

CS-3701 TRS-80 Disk, 48K \$24.95



Your object in this simulation is to be elected president. In your campaign you set your strategy and carry it out week by week. You may run TV or magazine ads, travel to different states, hold news conferences and participate in a debate.

You must take a position on ten campaign issues such as Energy Policy, Unemployment, Taxes, Mid-East Policy and Strategic Arms Limitations. You must manage your fund raising efforts to business, labor and mass direct mail solicitations.

The package includes four models of varying complexity, each can be used at ten levels of difficulty. The more complex models introduce the influences of incumbency, campaign finance and spending limits.

Hail to the Chief has been used as a teaching aid in Political Science, Voting Behavior and Computer Science at the University level since 1976. It is a well proven package which includes a comprehensive manual.

3 Adventures

Disk CS-3516 \$39.95
Requires 32K



Adventureland (by Scott Adams)

You'll encounter wild animals, dwarfs and many other puzzles and perils as you wander through an enchanted world, trying to rescue the 13 lost treasures. Can you rescue the Blue Ox from the quicksand? Or find your way out of the maze of pits? Happy Adventuring!

Pirate Adventure (by Scott Adams)

Yo Ho Ho and a bottle of rum. . . You'll meet up with the pirate and his daffy bird along with many strange sights as you attempt to go from your London flat to Treasure Island. Can you recover Long John Silver's lost treasures? Happy sailing matey!

Mission Impossible Adventure (by Scott Adams)

Good Morning. Your mission is to . . . and so it starts. Will you be able to complete your mission in time? Or is the world's first automated nuclear reactor doomed? This one's well named, it's hard, there is no magic but plenty of **suspense**.

Good Luck.

Voodoo Castle The Count and Ghost Town

Voodoo Castle (by Scott Adams). Count Cristo has had a fiendish curse put on him by his enemies. There he lies, you are his only hope . . . will you be able to rescue him—or is he forever doomed? Beware the Voodoo man.

Disk CS-3517 \$39.95
Requires 32K

The Count (by Scott Adams). You wake up in a large brass bed somewhere in Transylvania. Who are you, what are you doing here, and why did the postman deliver a bottle of blood? You'll love this Adventure. In fact, you might say it's Love at First Byte. . .

Ghost Town (by Scott Adams). Explore a deserted western mining town in search of 13 treasures. From rattlesnakes to runaway horses, this Adventure has them all! Just remember, pardner, they don't call them Ghost Towns for nothin'! (Also includes a new bonus scoring system.)



Original Adventure

Disk CS-3518 (48K) \$19.95

This is the original adventure game complete with a colossal cave populated with nasty little dwarves, a giant clam, trolls and much, much more. Includes the SAM76 language in which the game runs.

Adventures on Cassette

Five adventures are available separately on cassette. Each requires 16K and costs \$14.95.

- CS-3007 Adventureland
- CS-3008 Pirate Adventure
- CS-3009 Mission Impossible
- CS-3010 Voodoo Castle
- CS-3011 The Count

Order Today

To order any of these software packages, send payment plus \$2.00 postage and handling per order to Creative Computing, Morris Plains, NJ 07950. Visa, MasterCard and American Express orders may be called in toll-free.

Order today at no risk. If you are not completely satisfied, your money will be promptly and courteously refunded.

Creative Computing Software
Morris Plains, NJ 07950
Toll-free 800-631-8112
In NJ, 201-540-0445

creative computing software



Computer Graphics

by Joan R. Truckenbrod

Reflective Symmetry Pattern Systems

Patterns developed with reflective symmetry systems consist of a figure and its mirror image. A pair of images consisting of the original figure and its reflection or mirror image can be created by holding a picture perpendicular to a mirror. The image in the mirror appears to be backwards. A reflective symmetry pattern includes the figure in its original orientation and in its reflected or mirrored state. This pair of figures can be repeated and arranged on various types of grids. (See *SoftSide* June and July 1981 for descriptions of alternate grids used for creating patterns.) In reflective symmetry a figure is reflected in reference to an axis. This axis can be horizontal, vertical or diagonal. This article describes the process of reflecting a figure in relation to a vertical axis. The next issue will include an article on the use of a horizontal axis for reflection. The combination of horizontal and vertical reflections will be illustrated in the following issue.

The two figures you see when you hold a picture perpendicular to a mirror constitute a pair of reflected images. The axis of reflection is the vertical line on the mirror where the picture is touching the mirror. To see a second pair of reflected figures, place the opposite edge of the picture perpendicular to the mirror. A figure can be reflected to the right or to the left as the axis can be placed on either side of the figure. Two distinct symmetrical pairs can be created by placing the vertical axis to the right or left of the figure and reflecting the figure on the respective axis. Figure 1 illustrates the effect of placing this axis of reflection to the right of the figure and then to the left.

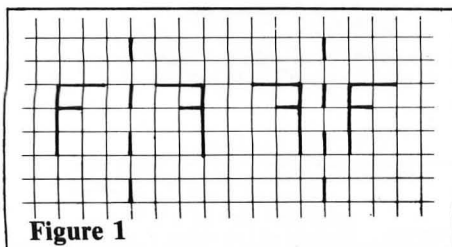


Figure 1

The reflection in these examples is reflection in a horizontal direction; consequently the X coordinates of the original figure are changed in creating the coordinates of the reflected figure.

In order to create the mirror image of a figure, all of the X coordinates of the original figure are subtracted from the largest X coordinate in the figure. The programs illustrated here use a design module 23 units wide, so all of the X coordinates are subtracted from 23 to create the X coordinates of the reflected figure. The Y coordinates remain the same since the figure is only being reflected towards the right or left and not up or down. The subroutine in the program that performs this reflection is subroutine 4000.

The reflexive pairs in Figure 1 can be repeated in different arrangements, as shown in Figure 2, to create reflective symmetry patterns.

The following programs can be used to create patterns based on both arrangements shown in Figures 2a and 2b. The first arrangement is created with the original program, Horizontal Reflection 1. Patterns in Figure 3 were created with this program. The figure arrangement in Figure 2b can be created by adding four program lines (107, 190, 195, 200) to the original program. This modified program, Horizontal Reflection 2, is illustrated in Figure 4. The design modules used in these programs are constructed in a square 23 units by 23 units. Experiment with your designs, as these programs can be used imaginatively to create dynamic patterns.

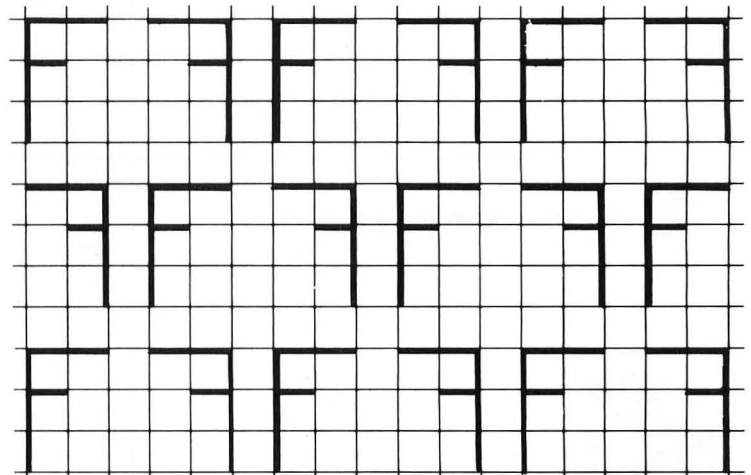


Figure 2a

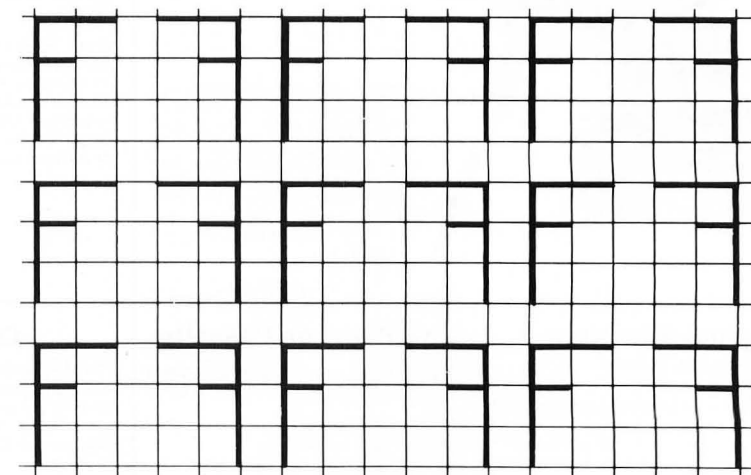


Figure 2b

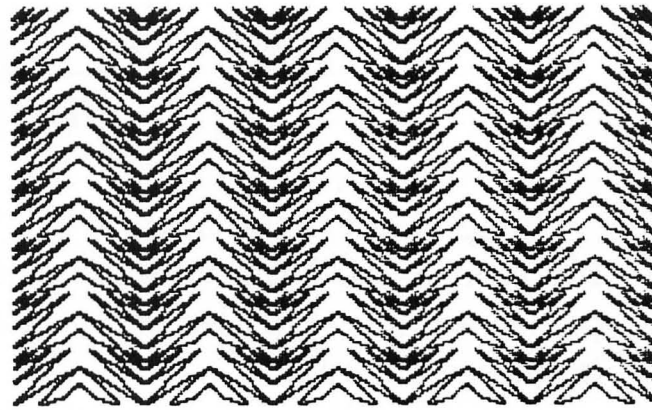
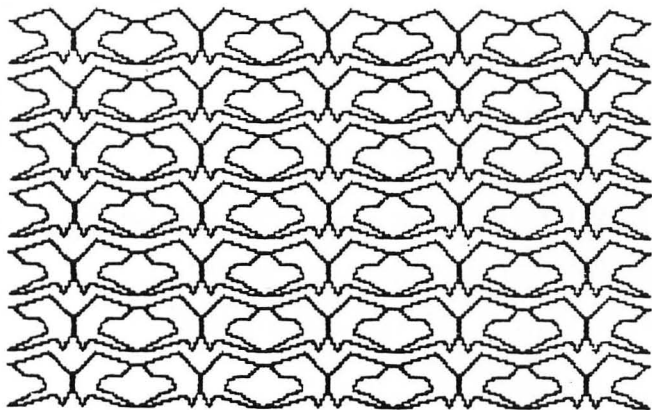


Figure 3

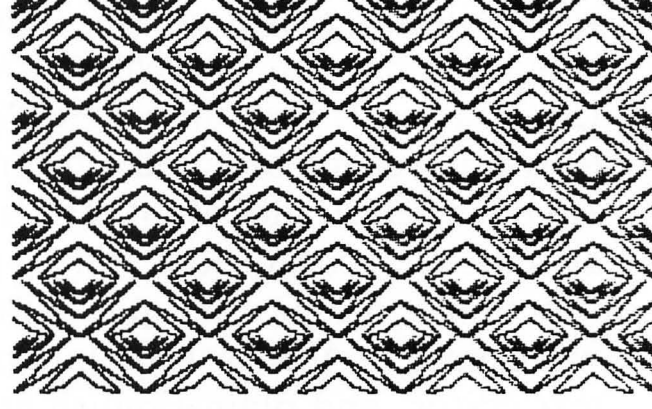
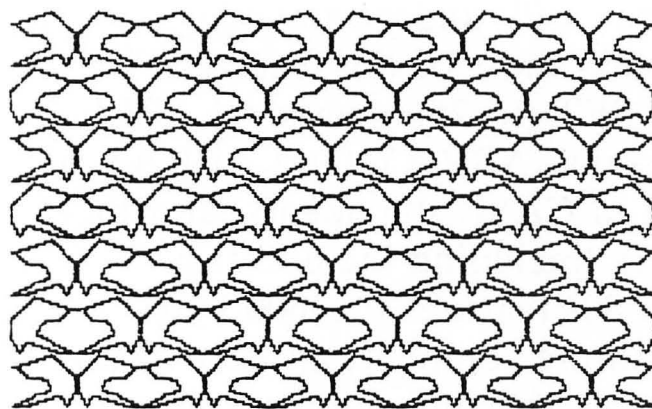


Figure 4

Horizontal Reflection 1

```

5 TEXT : HOME
10 REM HORIZONTAL REFLECTION 1
15 REM BY JOAN R. TRUCKENBROD
20 REM COPYRIGHT 1981
50 DIM X(50),Y(50),X1(50),Y1(50)
  ,NX(50),NY(50)
58 REM NPTS IS THE NUMBER OF
  POINTS IN THE FIGURE
60 NPTS = 12
70 FOR I = 1 TO NPTS: READ X(I),
  Y(I): NEXT I
90 REM DATA DESCRIBE THE FIGURE
95 DATA 15,0,23,10,23,17,20,23,
  18,19,6,23,0,23,14,15,11,10,
  6,10,0,5,15,0
103 GOSUB 4000
104 HGR2
105 HCOLOR= 7
109 L = 1
110 FOR N = 12 TO 160 STEP 24
120 FOR M = 5 TO 240 STEP 24
150 ON L GOSUB 5000,5100
160 GOSUB 3000
170 L = L + 1
175 IF L > 2 THEN L = 1

```

```

180 NEXT M
250 NEXT N
350 END
3000 REM PLOTTING SUBROUTINE
3005 IF N > 179 GOTO 350
3010 HPLLOT NX(1) + M,NY(1) + N
3020 FOR I = 2 TO NPTS
3030 HPLLOT TO NX(I) + M,NY(I) +
  N
3040 NEXT I
3050 RETURN
4000 REM SUBROUTINE FOR
  HORIZONTAL REFLECTION
4020 FOR I = 1 TO NPTS
4030 X1(I) = 23 - X(I)
4040 Y1(I) = Y(I)
4045 NEXT I
4150 RETURN
4900 REM SUBROUTINE 5000 ASSIGNS
  VALUES FOR PLOTTING ARRAYS
  NX AND NY
5000 FOR I = 1 TO NPTS
5010 NX(I) = X(I)
5020 NY(I) = Y(I)
5030 NEXT I

```

```

5040 RETURN
5100 FOR I = 1 TO NPTS
5110 NX(I) = X1(I)
5120 NY(I) = Y1(I) + .5
5130 NEXT I
5140 RETURN

```

Horizontal Reflection 2

```

10 REM HORIZONTAL REFLECTION 2
107 C = 1
109 L = 1
110 FOR N = 12 TO 160 STEP 24
120 FOR M = 5 TO 240 STEP 24
150 ON L GOSUB 5000,5100
160 GOSUB 3000
170 L = L + 1
175 IF L > 2 THEN L = 1
180 NEXT M
190 C = C + 1
195 IF C > 2 THEN C = 1
200 L = C
250 NEXT N
350 END

```

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



SoftSide
Selections

6 South Street Milford NH 03055
For Orders Only 603-673-0585

DOSPLUS

FEATURES:

- 1) Radio Shack compatibility
- 2) Error free variable length records
- 3) Full lower case detection and support
- 4) Repeating keyboard with NO keybounce EVER
- 5) Shift [0] typewriter keyboard option
- 6) Execute only protection feature for BASIC programs
- 7) Automatic track support for 35 through 80 track drives (mixed)
- 8) Device I/O handling with FORCE command
- 9) Supports high speed clock modification (up to 4.0mhz)
- 10) Supports mixed mode (single & double density) automatically
- 11) Allows disable-enable of break key
- 12) Allows user to define step rate per drive and re-configure system disk
- 13) Allows for efficient use of double-headed drives
- 14) Built in screen printer (shift [CLEAR]) with [BREAK] key abort
- 15) Multiple command chaining with "DO"
- 16) Built in memory test with CLEAR command
- 17) New printer driver which allows complete forms control and paging
- 18) Automatic serial printer driver with optional auto linefeed
- 19) Execute any DOS command from BASIC and return to BASIC
- 20) Free space map of diskette with optional output to printer
- 21) Copy with variable length files
- 22) Complete RS232 control from keyboard with status check
- 23) Create and pre-allocate files from DOS
- 24) Display current date and time from DOS
- 25) More information from Directory with optional printer output
- 26) Enter DEBUG with shift [BREAK] to allow use of [BREAK] from BASIC
- 27) New DISKDUMP/CMD sector display/modify program (works with filespecs)
- 28) New DISKZAP/CMD single/double density disk editor
- 29) New BACKUP (more reliable, no more pack ID check)
- 30) New FORMAT (more reliable, no need to bulk erase disk first)
- 31) New MAP utility (maps out disk, showing where files are located)

New DOSPLUS Z80 Extended Disk BASIC

- 1) Faster loads and saves
- 2) BASIC Reference utility (lines, variables, keywords, printer option)
- 3) BASIC Renumber utility (renumber section of text, block text mode)
- 4) Shorthand features for almost ANY direct command (LOAD, SAVE, etc.)
- 5) Shorthand features for editing (listing and editing with single key)
- 6) CMD "M" instantly displays currently set variables
- 7) Global search and replace in BASIC text
- 8) Line printer TAB to 255
- 9) OPEN "E" to end of sequential file (for output)
- 10) DI (delete and insert text line)
- 11) DU (duplicate text line)
- 12) ".R" & ".V" options after LOAD and RUN (files open & save variables)
- 13) OPEN "D" allowed (Model II compatible) equal to OPEN "R"
- 14) DOS commands from BASIC
- 15) Automatic, error-free variable length records
- 16) Single step execution with TRON (fabulous for debugging)
- 17) CRUNCH (BASIC program compressor)
- 18) New TBASIC (tiny BASIC) offers full BASIC commands
- 19) TBASIC and DOSPLUS together only use BK of RAM (40K left in 48K TRS-80)

***** 7 MORE UTILITIES *****

- 1) Single drive copy
- 2) Restore (dead files)
- 3) Purge (unwanted files)
- 4) Clearfile (destroys data by writing zeros to file)
- 5) Transfer (moves all user files from one disk to another)
- 6) Spooler (allows printing of text while freeing up the CPU)
- 7) Crunch (Basic program compressor)

DOSPLUS gives you more of what you buy an operating system for. Speed and reliability without sacrificing simplicity and power. If you need extra power without extra wait, then you need DOSPLUS!

Single or double density systems available for Model I. Model III DOSPLUS ready for immediate delivery.

Perhaps the best investment you can make for your TRS-80! Listen to what others have had to say about DOSPLUS.

"Overall, DOSPLUS is the fastest operating system I have seen..."

Pete Carr in 80-US Journal.

"DOSPLUS...the better mousetrap."

Stewart Fason in 80-Microcomputing

"On a scale of 1 to 10, I give DOSPLUS a solid 9."

Reese Fowler in 80-Microcomputing
(Model III DOSPLUS review)

For the BASIC programmer, our features are unmatched. For the average businessman, our speed and simplicity cannot be beat.

So, join the satisfied users who have joined DOSPLUS. Experience excellence! Experience DOSPLUS!

Model I DOSPLUS

#25-217001D \$99.95

Model I Double Density DOSPLUS

#25-217002D \$99.95

Model III DOSPLUS

#27-217003D \$99.95

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



Video Reverse Modification

by Edward E. Umlor

The aim of this article is to give you a very inexpensive way to obtain reverse video on your S-80. I have installed and used other (keyboard activated) reverse video modifications and had some irritating results. When I sat down to look into the main problem created by most modifications, the cause of it became clear. This mod takes care of the problem, and so far has not created any of its own.

The problem: Popping in and out of reverse video as you are inputting data or a program.

The cause: Typing too fast for the rollover and striking multiple keys at the same time.

The result: The rollover buffer receives sufficient strikes to generate a trigger code (turning the video reverse on or off) before being cleared.

Due to the level of my gray matter, I am a firm believer in the KISS principle (Keep It Simple, Stupid). The first question I asked myself was: How do we make it really on or off and not in-

between? The answer was a simple Single Pole, Double Throw (SPDT) switch. In one position the video is normal and in the other it is reversed. Well, that took care of the blink problem.

The next question was: How do we actually reverse the video? The answer was again quite simple — invert the signal before sending it out to the video. This is accomplished by sending it through a NAND gate before the final output stage. This changes the level of the background to the ON values, and the level of the characters to the OFF values. The point in the circuit that was chosen for the modification is the composite video (alphanumeric + graphics) just before the stage where the sync signals are added to form the final video signal. That way the syncs are not affected and video stability is maintained. The picture is produced by accelerating electrons to the phosphor-coated face of the tube. When the electrons strike, the phosphor gives off

light. The ON value allows electrons to flow and the OFF value closes the gate to the electrons.

Everything was okay except for a leaning effect caused by reversing the video. A full line of CHR\$(191) — full-pixel graphics character — would cause a lean to the right of about 45 degrees. A very simple compensation circuit installed in the video monitor takes care of that problem. Other modifications also require a compensation circuit, as the problem is a function of the video itself. A capacitor, three resistors, one transistor, and a one-inch square of perforated board (.1" vector board) is all that is needed for the circuit. Some wire, a piece of foam, and a piece of electrical tape will get it mounted and installed in the video.

That, in short, is the modification and logic behind it. I wanted to do the easiest possible modification consistent with stable video operation. To the best of my knowledge, the mod presented here is the simplest and

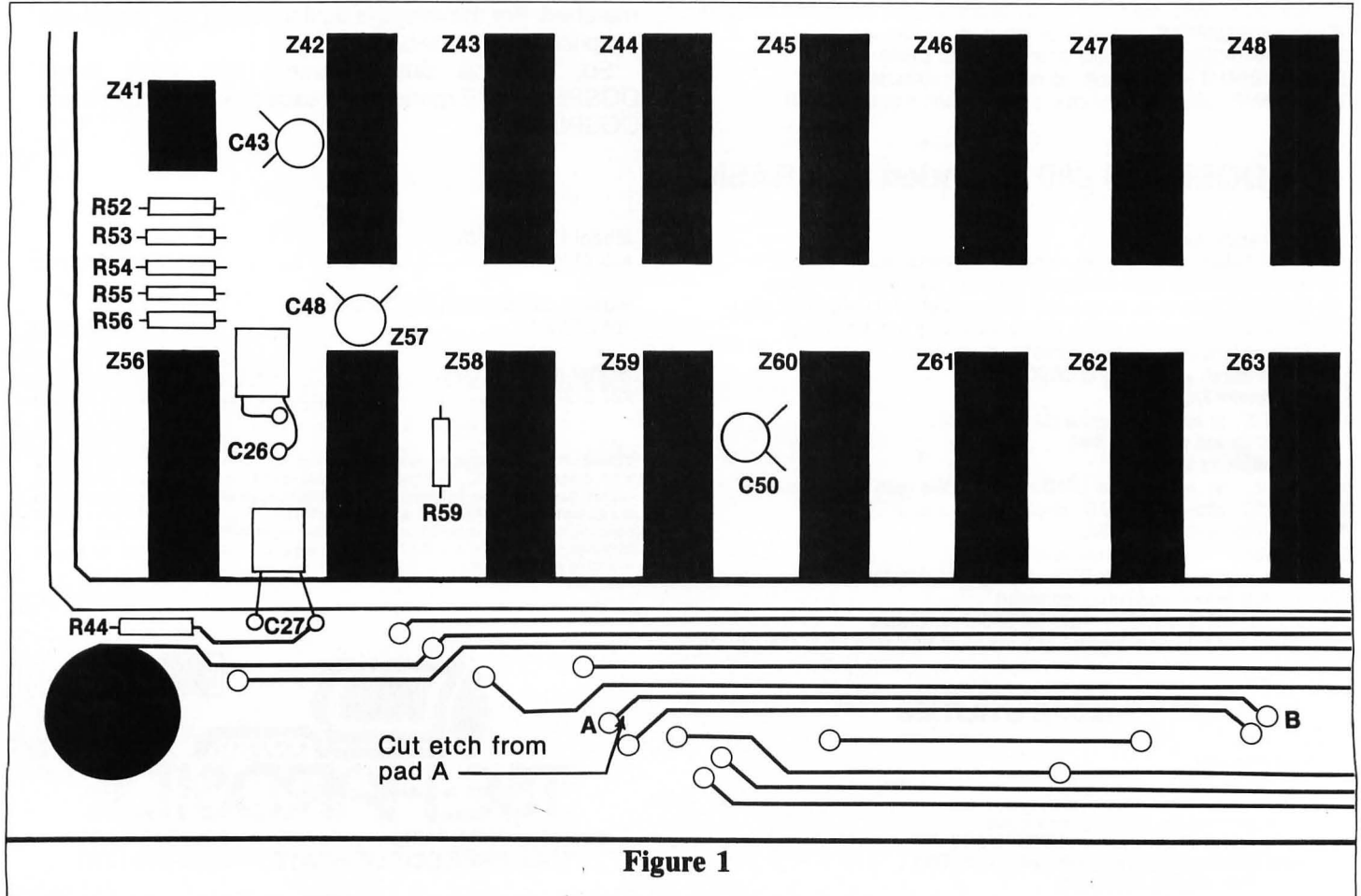


Figure 1

easiest to install of any. So let's get into the meat of the thing and start off with the parts list:

- One 74LS00 dual input quad nand gate.
- One SPDT switch, single pole double throw subminiature.
- Three feet AWG#26, insulated, solid hookup wire.
- One general purpose NPN small signal transistor (2N3904 or equivalent).
- One 4.7 microfarad, 16 WV, tantalum capacitor.
- One 470 ohm, ¼ watt, resistor.
- One 1.5 Kohm, ¼ watt, resistor.
- One 470 Kohm, ¼ watt, resistor.
- One square inch .1" on center vector board.
- One square inch insulating foam ¼ to ½ inch thick.

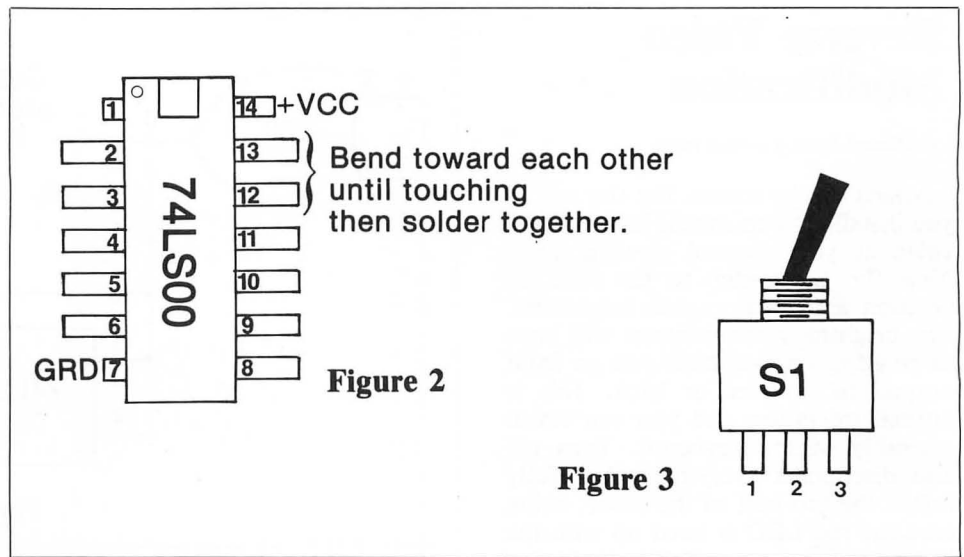
- Two square inches of double sided adhesive tape (or tape in position with electrical tape).

The total cost of this parts list should be in the neighborhood of \$5.00. The source of your supplies is strictly up to you and shopping around can save you up to 50% of the list price.

The tools you will need are:

- One Phillips #1 screwdriver.
- One Xacto or other sharp knife.
- One pair of small diagonal cutters.
- One pair of small longnose pliers.
- One soldering iron; 25 watts is sufficient. (Don't want to burn the board.)
- One pair of wire strippers or good set of teeth.
- One of rosin core electrical solder. AWG#22 wire size is good. (We don't want to solderpot the whole circuit.)
- One damp sponge to keep the tip of the soldering iron clean.
- One small adjustable wrench.
- One ¼ inch nutdriver.

Let's do the keyboard changes first. Disconnect all cables from the keyboard and remove them from your system. Now go hide in your workshop so the wife won't see what you are about to do to that thing you bought with the dishwasher money. Turn the keyboard over and remove the six screws from the bottom with a Phillips screwdriver. Press your thumb over the label on one of the screws. It will recess over the screw hole and can be removed by using the Xacto knife to cut around the edge of the hole. **BYE-BYE** to the warranty. However, by now, all Tandy warranties on the keyboards are expired since they haven't manufactured any in 1981. Turn the keyboard over and gently lift off the top cover. This will expose the keyboard and the etch side of the logic board. Gently lift up the keyboard, tilting it toward you



(do not put strain on the ribbon cable connecting the two boards), and remove the five white insulating spacers. Lay the keyboard back onto the logic board, grasp the edges of the logic board (large board below the keyboard), and carefully remove it from the case. Set the case aside for the moment. Turn the assembly over again to expose the component side of the logic board.

You will now have to decide where you want the switch. I personally do not like to put it on the case as it makes the removal of the case more difficult. I prefer to mount the switch on the black faceplate that protects the DIN connectors and power switch. This faceplate is removable, which simplifies the drilling and lessens the danger of damage to the keyboard. The switches will mount in a hole ¼ inch in diameter, and I recommend drilling the hole just above the power-switch hole (when the keyboard is reassembled it will be below the power switch). The switch is mounted horizontally (parallel to the long edge) in the hole and be sure to run the nut down tight. Replace the face plate and make sure the switch body does not short out (touch) any of the power-switch contacts. So far, so good!

Place the keyboard in front of you with the ribbon cable toward you on the right side. Refer to Figure 1 for the following: Locate Z59, then pad A, and then, with the Xacto knife, cut the etch leaving pad A. Be very careful to cut only the one etch; any others will have to be repaired. Prepare your 74LS00 as shown in Figure 2. Pins 1, 14, and 7 will be soldered to the chip used for piggybacking. Locate Z42 and solder the 74LS00 on top of it. Strip one end of the wire, locate pad B, and solder the wire to it. Run the wire to the 74LS00 pins 12 and 13, and cut the

wire off long enough to allow you to strip and solder the wire to these pins. Strip the wire and hook it onto the two pins. Strip one end of the remaining wire, hook it onto the pins and solder. Run the wire to pin 1 (see Figure 3) of the switch, cut off, strip, and solder to S-1. Strip one end of the wire, solder to 74LS00 pin 11, run to the switch pin 3, cut, strip and solder to S-3. Locate pad A and connect to switch pin 2 in the same manner. You have now completed the keyboard wiring and the schematic in Figure 4 is now the correct road map. If you have a tech manual, page 108 should be corrected. You will also have a little more than a foot of wire left, and it will be used to wire the compensation board into the video.

Time to do a final inspection of your work for solder bridges (the unintentional soldering together of conductors or pins), unsoldered joints, and routing of the wire so as not to interfere with assembly. **EVERYTHING IS OK!!! FANTASTIC!!!** Now, let's put it all back together. Turn the logic board over, being careful of the ribbon cable, and set it aside within easy reach. Place the bottom half of the case in front of you with the cutouts toward the back. Be sure the white spacer (rigid plastic) is on the post on the right side. Gently lower the logic board back into the case, making sure it is fully seated. The switch you have installed might interfere with the bottom of the cutout. If it does, cut out the bottom of the cutout with the Xacto knife or file until proper clearance is obtained. Replace the five insulating spacers, two on top, two on bottom, and one on right side. Before we close up the keyboard, let's see if it is working. Hook up the power cable and the video cable to the keyboard. Turn on the video and then the keyboard. If you can hardly see

continued on next page

Reverse Video Modification

continued from previous page

anything on the screen, flip the switch you installed. You should have normal video at your normal viewing level. Now flip the switch to the reversed position and turn up your brightness. The brightness and contrast will have to be adjusted each time you go from normal to reversed or back. This is normal operation and you can finish assembly of the keyboard. Turn off and disconnect everything. Carefully install the top half of the cover, being sure the red LED is lined up with the hole in the cover.

Turn them over and, holding the two halves together with one hand, find a middle-length screw, insert in the side hole, and tighten it with a screwdriver until snug. You can let go now and install the rest of the screws. The short ones go to the front, middle length to the sides, and the longest ones to the back. Snug the screw only; you are working with plastic and the screw holes will strip out easily.

If you are only going to use your system for business or word processing and not run any games with graphics, you can stop right here. However, if you are going to do anything else, you will have to compensate your video. First you make your little circuit on the one square inch of vector board. One suggested layout is seen in Figure 5. Insert the transistor into the board and bend the legs in the direction you want them to hook up to the other components. There should be enough lead length to keep you from using any wire. Preform your components and insert them into the board. You might want to piggyback the capacitor on the 470K ohm resistor before assembling. This is the area where you can be very creative — just be sure to observe the polarity of the capacitor and the base, emitter, and collector of the transistor. See Figure 6 for a schematic of the circuit. Once you have completed this little task you are ready to play the mad scientist again. Sneak your video into your operating room and prepare yourself to operate. Place the video face down on the operating table and using the 1/4 inch nutdriver, remove the five screws that secure the back cover. There are four of them recessed at each corner, and one right out in the open by the power cord. Gently lift the cover off, feeding the power cord through

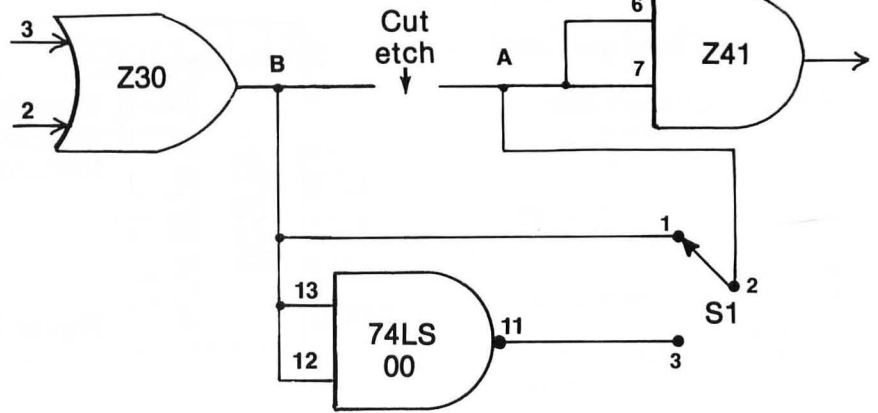


Figure 4

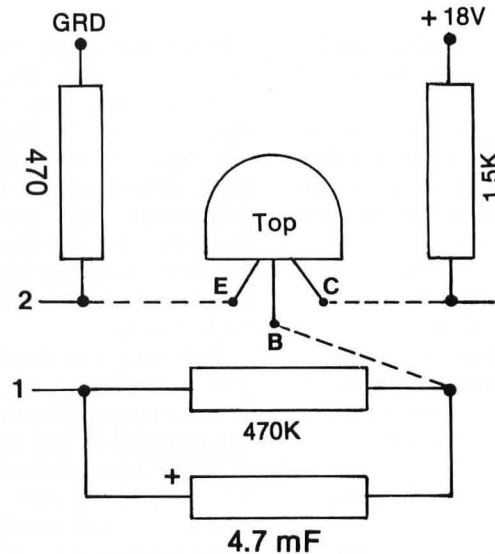


Figure 5

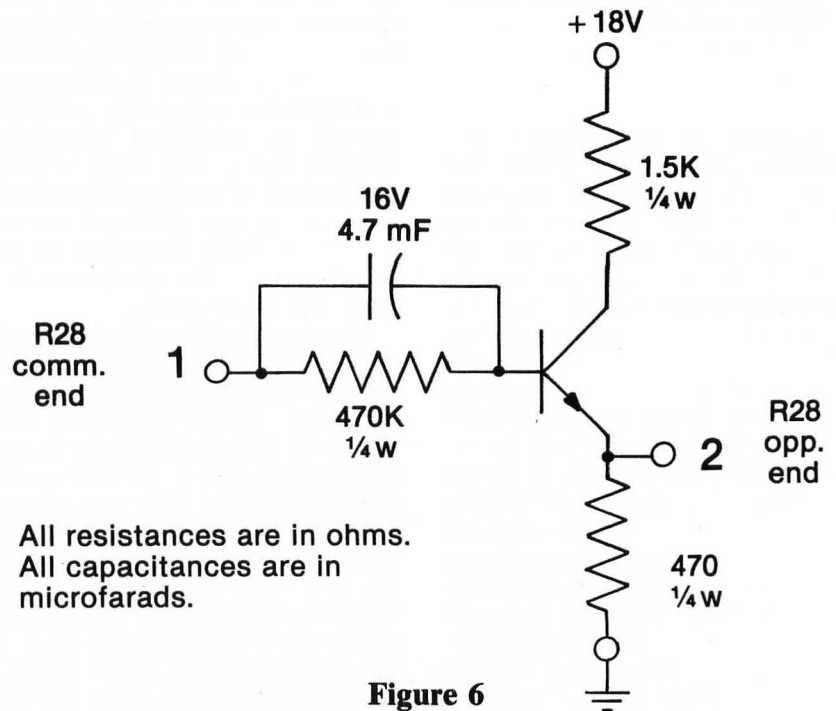


Figure 6

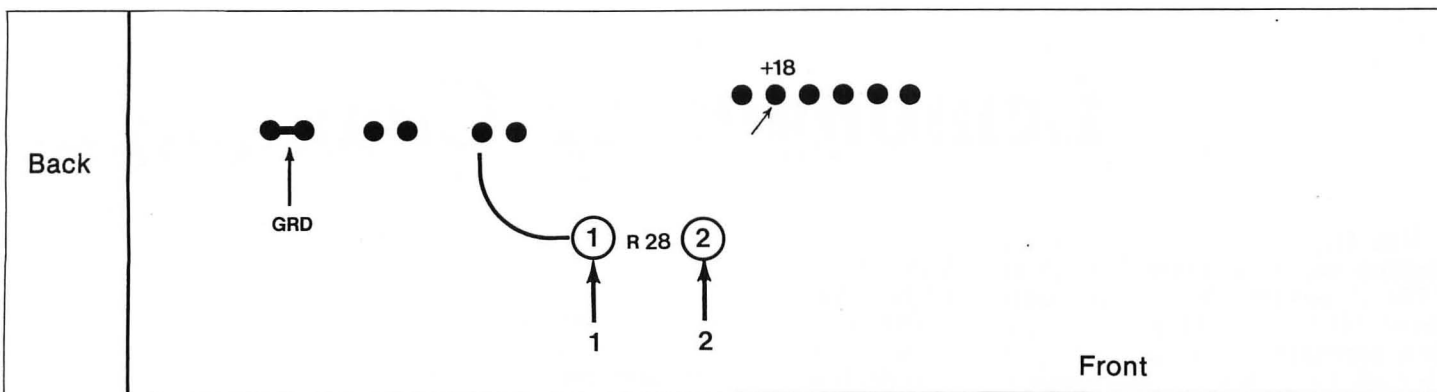


Figure 7

the hole, and set the back cover off to one side. Use Figure 7 for the installation of the compensation board. Now set the video on its top with the face away from you. Gently pull the circuit chassis toward you to expose the etch side of the circuit board, until you can reach the connections needed. Lower the back until it supports itself on the chassis guides. These guides are plastic and will not stand a lot of stress.

It is easier to do these next steps without the compensation board being permanently mounted. Place the board in the position you will use permanently. Measure, cut, and strip the wires for +18V, ground, and pads 1 and 2. Solder these onto the compensation

board. Now locate and remove R28 from the video board. Mount the compensation board permanently on the video circuit board and solder the four wires to their proper location. Lift the back of the chassis and gently push it back into its proper position in the video unit. This completes the wiring required for the video compensation. Place the video back on its face, feed the power cable through the proper hole and seat the back cover in correct position. Replace the five screws and snug them. Once again you are working with a soft plastic, so be careful not to strip out the screw holes. NOW IT IS TIME TO CLEAN UP THE OPERATING ROOM AND TAKE

YOUR BOWS. YOU HAVE COMPLETED THE MODIFICATION!

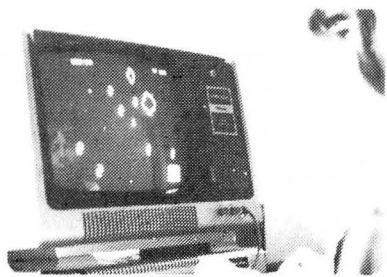
Set up your computer system again and check to see that everything operates properly. Print a couple of lines of CHR\$(191) to be sure the lean is out. Some of the older videos do not have as much contrast as the newer ones. This is something other than compensation and the video will have to be adjusted.

And now as the great video in the sky sinks slowly in the west, your ole buddy bids you adieu until the next great spurt of incredibly dump genius. SO LONG AND BOY IS "METEOR MISSION II" GREAT IN REVERSE VIDEO.

Games from **BIG FIVE** will turn your computer into a

TRS-80TM HOME ARCADE

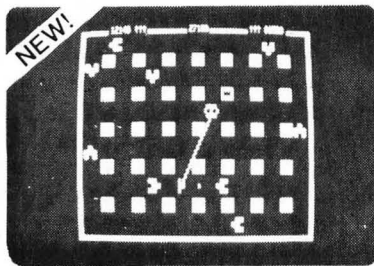
SUPER NOVA[©]



"Huge ASTEROIDS have invaded the galaxy! Your mission is to destroy them and the alien saucers before they destroy you!" *Our #1 top selling game!*

\$15.95 16K Mod I or III

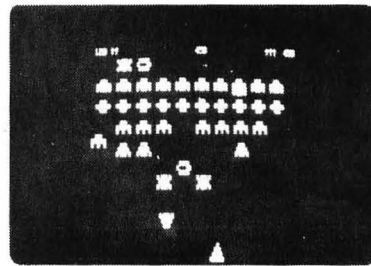
ATTACK FORCE[©]



"Eight alien ramships are warping down toward your destroyer ship. You must shoot them down quickly before they crush you!" *With sound!*

\$15.95 16K Mod I or III

GALAXY INVASION[©]



"The newest and most exciting Invaders-type game yet! Smooth sound effects, sharp graphics, and the 'Flagship' alien from **Super Nova** combine to make this our finest TRS-80 game!"

\$15.95 16K Mod I or III



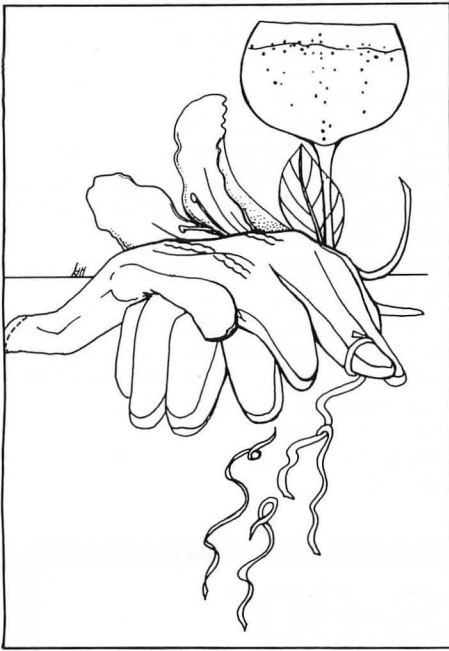
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790





Lemonade or Champagne

Will Hagenbuch is the author of business and utility programs such as "File Manager 80", "Accounts Receivable", and "Utility". This is the final installment of our serialization of his book *Lemonade or Champagne*, a guide to business software.



by Will Hagenbuch

Trust me that, as we proceed through this and the next sections, you will find out that this information on how to "OPEN" a random file is really all you need to know! What we will be doing, in this Section and the next, is introducing you to File Manager-80, an effective alternative for disk I/O handling. However, before we get involved with File Manager-80, let's discuss some of the other aspects of systems development using disk files.

RANDOM ACCESSING

Let's assume for the moment that you have, or will have, the ability to use random accessing techniques to fetch and write records from and to a file. It would naturally follow that you would need to know "which" record. This is quite easy if you have a listing of the file and it contains the number of each record, but what if you don't have such a listing? Well, you might consider accessing every record on the file with a FOR . . . NEXT loop and inspecting the data content of the file to

see if that is the one you wanted. However, this is no more efficient than a Sequential File, is it?

The determination of "which" record is best done by the use of an Index. An Index might be described as a pointer to "which" record on your random access file contains the information that you want. There are two methods of creating such an Index: either by using the FOR . . . NEXT loop to read all of the records from your random file one time, or by creating and maintaining a separate "Index" file (which is normally a Sequential File). In either case, the file which is the basis of your Index will be read into a memory array which we will call your Index Table. Naturally, you must have included a dimensioned array in your program to accommodate this Index Table which must include, as a minimum, the Data Element which is to be your "key" for locating your record. This "key" may be either an alphanumeric or numeric element and you must dimension accordingly — and you will need a sufficient array size so that there is one bucket for every record in your file, including any file size expansion that you may have in the future!

As an example, let's say that you have a random file of customers. At the time each of these customers was placed in the file, a unique number was assigned to each customer. This Customer Number is the Data Element that will be used as the "key" for accessing the customer record. If the Customer Number is a part of the customer file it will need to be split off (using the FOR . . . NEXT loop) to create an Index Table each time a program requiring random access of the file is run. Similarly, if the Customer Numbers had been maintained in a separate file they would also be required to be accessed each time the program is run, but this would appear to be the more prudent way since fewer bytes of data would need to be accessed in order to build the Index Table. In either case you would place the Customer Number into a one-element array that had been dimensioned to accept it.

Now that the Index Table of Customer Numbers has been established, it is a simple procedure to relate the Customer Number to the physical record location for that customer's

record. By searching the table with the desired Customer Number and finding a match (equal comparison) on the Nth record, we know that the customer record is the Nth record in our random access file.

The searching of the Index Table is, however, an art unto itself. If we were to insure in our program which creates customer records, that all Customer Numbers were assigned sequentially in ascending sequence, we could use a Binary Search to locate the desired Customer Number very quickly. A Binary Search is a table search which "bisects" the table entries. In other words, if we were to compare the desired Customer Number to the "middle" entry in the Index Table of customer numbers and that comparison told us that the desired number was less than the middle entry, we would have effectively reduced further searching by one-half. Obviously, the Customer Numbers between the middle and the top (last number) of the array will not match the desired number. So, let's look at the bottom half of the Index Table. See Figure 3-1.

Our next comparison would be at the first Index Table entry (or the last entry if the key we are searching was "high" to the middle). If our key is "low" to the first entry (or "high" to the last entry) we know immediately that the number we seek is not on the table. Obviously, whenever an "equal" condition occurs, we have found our number and, consequently, our pointer for random accessing of the customer file.

If the second comparison (the one to the bottom of the Index Table) was "high", we know that our key number ranges somewhere between the first and middle numbers on the Index Table. Therefore, we would compute the middle location between the first and middle Index Table entries and do the third comparison against that entry. Again, if unmatched we would continue the bisecting until the number was either found by an equal comparison or determined to not be on the Index Table. The "not found" condition is determined when no more bisecting is possible and an equal condition has not been found.

Figure 3-1 provides an example of a Binary Search routine which might be included as a program subroutine. Follow the code in this example so that

```

400: 'Binary Search Subroutine
Sets SW=0 if "Found" else
SW=1 if "Not Found"
Variable "SR" is Search Key
Variable "T" is total number
of entries on table
Variable "LN" must have
previously been dimensioned
Variable "XL" is "Low" posi-
tion of table to be searched
Variable "XH" is "High"
position of table to be searched.
Variable "XM" is computed
"Middle" of table.
410: SW=0: XL=1: XH=T: IF
SR< LN(XL) OR
SR> LN(XH) THEN 470
'Set up search controls and exit
if key not on table
420: IF SR=LN(XL) OR
SR=LN(XH) THEN 460
'Exit if key matches first or last
entry
430: XM=INT((XH+XL)/2):
IF XM=XL OR XM=XH
THEN 470
'bisect remaining portion of
table and exit if table exhausted.
440: IF SR< LN(XM) THEN
XH=XM: GOTO 430
'Set control to continue search
in "lower" part of table
450 IF SR>LN(XM) THEN
XL=XM: GOTO 430
'Set controls to continue search
in "upper" part of table
460: RETURN 'Exit with
"FOUND" condition
470: SW=1: RETURN 'Exit
with "NOT FOUND" condition

```

FIGURE 3-1

you thoroughly understand its implication. Your search will work extremely fast with this searching method because you will be looking, in the worst case, at only about one-eighth of the Index Table.

However, if your Index Table was not in sequence, you could not use the Binary Search and would be forced to use the slower method — the Sequential Search.

In the Sequential Search, we must look at every entry in our Index Table, or at least until we find our match. We do this with the FOR...NEXT loop routine. The fastest method of performing this search is to break out when (and if) a match is found. If you are going to employ the "break-out" search loop, be sure that it is not a nested FOR...NEXT loop (inside of another FOR...NEXT loop) since you will leave an "unsatisfied NEXT" and that's a "no-no". An example of the "break-out" Sequential Search is provided as Figure 3-2.

```

400: 'Sequential Search
Subroutine
Variable "I" contains
"Found" location or zero of
"Not Found"
Variable "T" is maximum
number of table entries
Variable "SR" contains
Search Key
Variable "LN" is previously
dimensioned array which con-

```

FIGURE 3-2

Of course, in either case, your program will have to handle the situation of a return with a "not found" condition as indicated by a zero value in the Index Table pointer; "XM" in Figure 3-1 or "I" in Figure 3-2.

OPTIONAL FILE STRUCTURES

To get the maximum use out of your disk storage space, you should have a working knowledge of how you might organize files to best serve your needs. It often happens that if you are limiting yourself to a Fixed Format file structure, you might need to create several files, and consequently require several buffer allocations, when one Multi-Format file would do the job.

A Multi-Format file, for our purposes, is a file that contains more than one format of records, or records which have more than one format. As an example, you may have reason to require that a file contains a "header" record of one format and some variable number of "trailer" records in another format. We will be using the term "header" to define any format which is the first of a group of related records; the term "trailer" will refer to the remainder of the records in that group. In this example, we will refer to the structure, as a "Dual Format" file. Structure A, shown as Figure 3-3, provides a pictorial layout of the "Dual Format" file structure.

```

tains table entries
410: FOR I=1 TO T 'Set loop to
search entire table
420: IF SR=LN(I) THEN
RETURN 'Exit if "Found"
Variable "I" contains
"Found" location
430 NEXT I 'Keep looking until
table exhausted
440 I=0 : RETURN 'Set "I"
zero if "Not Found"

```

In Figure 3-3, we see a typical application of transaction entry employing the "Dual Format" file structure concept. The first record on the file contains such one-time information as the Transaction Batch Number, Date of the Batch, and Identification of the Operator who entered the information. This format appears only one time on the file.

The second format is that of the transactions which make up the batch. One record is used for each transaction and the number of this transaction format that may be placed on the file is limited only by the physical size of your recording media (disk or tape).

The second example of Multi-Formatted records will be called the "Variable Format". In this type of structure, we use two different formats in the same record; and, of course, the number of records that can be stored in a file is limited only by your physical storage capacity. Figure 3-4 provides a pictorial layout of Structure B, the "Variable Format" record.

In the "Variable Format" structure, the number of trailers for each header must be specified. Because both the header and the specified number of trailers must be contained in a single sector of disk (255 or 256 bytes, depending on the DOS you are using), the aggregate size (bytes) of the header plus the number of specified trailers

continued on next page

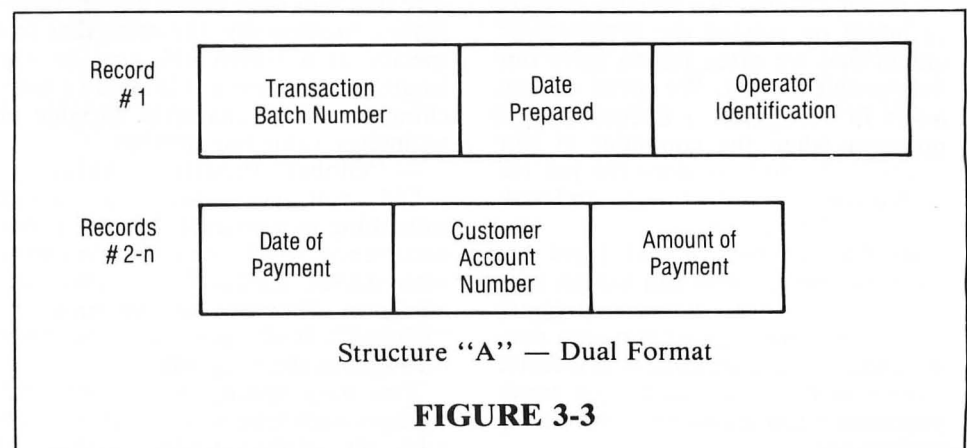


FIGURE 3-3

Lemonade or Champagne

Continued from previous page

cannot exceed the physical restriction placed upon you or your DOS.

In Figure 3-4, we see a typical application of transaction detail (trailer records) applied to a customer record (header). In this case, we have established a maximum of three detail records per header record. It could have been more or less depending upon our requirements — as long as the aggregate size of the header and the maximum number of trailers for that header does not exceed the physical block size (255 or 256, depending on the DOS we use). Note that this file structure is called “Variable Format” because the trailers may or may not contain information. This does not mean that unused trailers do not take up space — they do. But, whether or not they actually contain information is a matter for your program to determine.

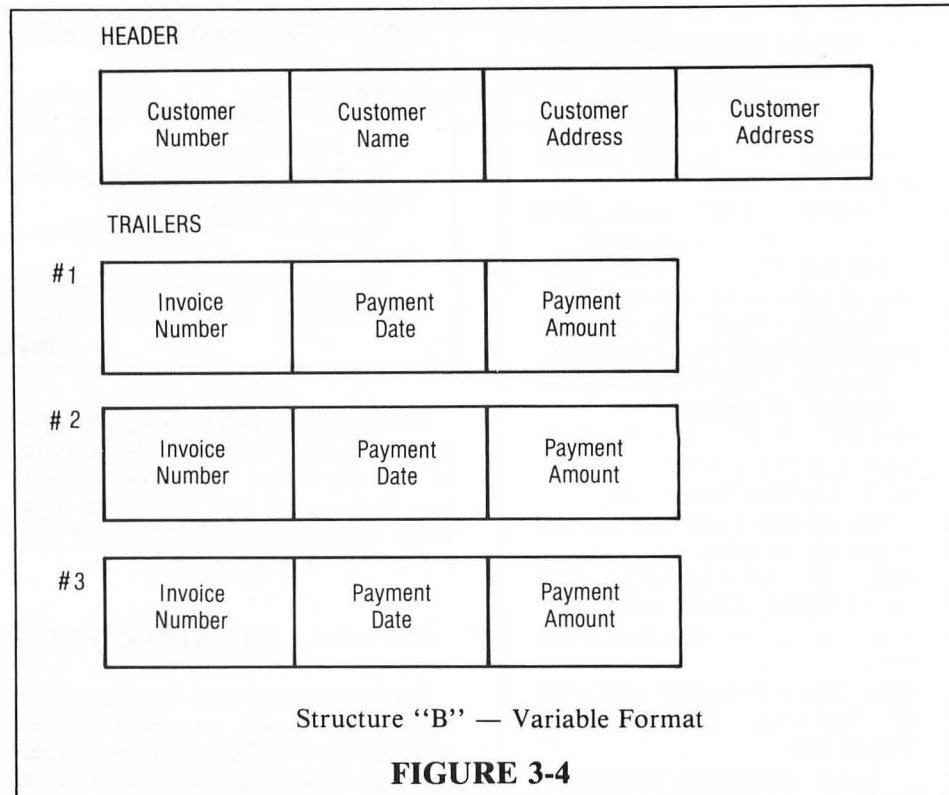
It might be noted, also, that in order to optimize file storage space, if the aggregate records size is less than one-half of the block size (255 or 256) the block may contain two (or more) of these “Variable Format” structure records. However, this “blocking” of records is only rhetorical since, if you are using File Manager-80, this is an automatic feature.

SECTION IV SOFTWARE SOLUTIONS

In the first three sections we described the magnitude of the effort necessary to professionally develop a microcomputer system. In this final Section we could like to offer some “software solutions” to assist you in accomplishing these tasks.

Like the proverbial barber who needed a haircut, the programmer often fails to utilize the power of the computer for his own benefit. We are so intent on solving the problems of others that we often fail to solve our own problems first. We spend endless hours in debugging or documenting a program when the computer at our fingertips could have done the job for us in a fraction of the time — and with guaranteed accuracy.

Would you believe that there are those among us who still use an old-fashioned typewriter to prepare systems documentation? Why not consider using your computer — it is faster and easier to use if you have a word-processor / text-editor program that is easy to use.



Structure “B” — Variable Format

FIGURE 3-4

PROGRAMMING AIDS

There are many ways that your computer can assist your programming efforts. We have put five of these “Utility” functions into one program. These utilities include Program Search, “Global” Program Modification, Line Number Cross-Reference, Edited Program Listing, and Program Compression.

— Program Search

This Utility will search your program for every occurrence of any string of alphanumeric characters you specify. It will then provide you with the line numbers of those programs lines which contain that string of characters.

Any of you who have ever searched through a lengthy program listing looking for some illusive piece of code know what a time-consuming battle this can be — and you are never quite sure that you have found all the occurrences. Incidentally, the string that you specify as a search key may be any length, so you are not limited to searching for a two-character variable or an integer value line number.

— “Global” Program Modification

Did you ever want to change something in a program? Not just one occurrence, which you could do simply with BASIC EDITOR, but every occurrence. This is what we mean by “Global”. It changes every occurrence throughout the program.

You may specify up to ten such changes each time you run this Utility and all alphanumeric strings of

characters that you specify will be changed to whatever other strings of characters you specify (or to “null” if you do not specify a string). The length of either the “FROM” or “TO” specifications is immaterial.

This can be a “dream” Utility if you should need to change something throughout a program — like a variable name, for example.

— Line Number Cross-Reference

Did you ever find yourself tracing through the logic of a program and say to yourself, “I wonder how it got there?” This Utility will answer that question for you by selecting each line containing a “GOTO”, “GOSUB”, “NEXT”, “ELSE”, or “RESUME” that causes a program transfer to another line and printing or displaying on the video screen (your option) the program line and each of the program transfer line numbers.

— Edited Program Listing

I don’t know about you, but my programs are not a very pretty sight after debugging is completed. I tend to make rather long code lines, especially where lines contain compounded “IF” statements. I also have the habit of throwing in comments (remarks) in a rather haphazard manner. To me, this is quite natural because, when I am coding, the logic is of utmost importance, not the structure. However, when I have completed the program, I sometimes want (or need) a “pretty” listing.

This Utility module will make structured listings of your otherwise messy code. All remarks will be set off to the right side of the listing, all subroutines will be so identified, spacing lines will be inserted between program segments, and instruction lines will be broken down to one instruction per print line. With this module, your program will not be altered.

— Program Compression

What the Edited Program Listing module does, the Program Compression module un-does. Program Compression will “crunch” your program into the minimum memory requirement by stripping out all remarks, all blank spaces, all “NEXT” variables and all line-terminating quote marks and then squeeze as many instructions into one line as possible.

Of course, the Compression module is smart enough to not alter the contents of strings and to preserve line numbers that are “addressed” elsewhere in your program, including “REMARKS” lines you may have addressed. But it will otherwise strip the program to its barest essentials to reduce memory requirements, reduce LOAD and RUN times, and make it a “bear” for someone else to modify and claim ownership!

By the way, a program which has been Compacted with this module can be expanded with the Edited Program Listing module, at the expense, of course, of all REMARKS statements and blank spaces which have been lost.

FILE MANAGER-80

If your Disk Operating System/Disk BASIC manual leaves you cold in its discussion of Disk Files, then you may want to consider File Manager-80 as an alternative way to create, code, and document Disk Files. The reason is that File Manager-80 is an Input/Output Control System (IOCS) that can

make those file structuring and random accessing problems go away because it writes your file Input/Output instructions for you! And, as a by-product, will produce a Dictionary of Data Elements and Record Layouts that are so vital to your system support documentation.

We have previously discussed several file structures (Dual Format and Variable Format) that might be employed when a plain vanilla Fixed Format record is not suitable. File Manager-80 will accommodate all three of these file structures. In addition, your worries (if you have any) about optimum record blocking and those pesky little algorithms you need to find sub-records are taken care of automatically by File Manager-80.

In summary, File Manager-80 is an IOCS for the S-80 that will make pages 7-37 through 7-75 of your Disk Operating System Manual go away. As we said earlier, if this part of the Manual left you cold, you might find that File Manager-80 is the solution. Or, even if you have mastered file accessing, File Manager-80 may appeal to you as a time-saving programming and documentation tool.

DOCU-WRITER

Docu-Writer is a text-editor and word processor designed for use by the programmer. Why? Because text-editing is performed in the BASIC EDITOR, an Editor familiar to BASIC programmers. In addition, Docu-Writer is a complete “system” with its own initialization program which enables you to “tailor” the program to your particular printer, memory capacity, and number of disk drives.

Docu-Writer performs the word processor functions with only five control codes; “#”, “<””, “C”, “@”, and the up-arrow. With only these few control codes, Docu-Writer is indeed one of the easiest-to-use word processors available that still provides a

full range of operations such as hyphenation, right-margin justification, line centering, page titling and numbering, variable line width and page length, “global” text modification, variable data insertion in text at print time, and UPPER/LOWER CASE WITHOUT HARDWARE MODIFICATION (provided your printer can print upper/lower case.)

Docu-Writer has one built-in shortcoming. It will not automatically write your program and systems documentation for you (you still have to hit the keys), but with all of this versatility, it will sure make your job easier. And, by the way, if by chance you write for profit, Docu-Writer will keep track of the number of words in your text files and make this information available to you.

SUMMARY

In this series we have attempted to do several things — not the least of which was to make you cognizant of the versatility and usability of File Manager-80, Docu-Writer, and Nepenthe Utilities as development and documentation tools for the System Developer. In addition, we have:

- Introduced you to the magnitude of the problems in “professionally” developing a computer system — micro or otherwise;

- Offered some insight into the methodology of creating “lasting” support documentation for computer applications;

- Covered, albiet cursory, the accessing of random file records by a couple of techniques;

- Provided a couple of alternatives to the plain vanilla Fixed Format file structure;

- Described a viable alternative to the “ho-hum” of writing file Input/Output instructions and documentation; and,

- Presented, for your consideration, several “software solutions” available to the System Developer. ☺

Atari One Liners

```
1 GRAPHICS 19:S=RND(1)*234:POKE 708,S
:COLOR 1:PLOT S/6,0:DRAWTO S/6,19:SOB
ND 0,S,12,10:FOR L=1 TO 45:NEXT L:RUN
```

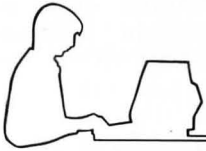
David Simmons
Redondo Beach, CA



Diskettes for Apple II and TRS-80

1 to 5 diskettes only \$2.50 each
5 or more only \$2.25 each

Write to: **Simon Warner**
1364 Grant Street
Lincoln Park, MI 48146



VARPTR Used

by John T. Phillip, M.D.

The second of three articles on the S-80 VARPTR function.

SUPER GRAPHICS

Now that we have VARPTR under control, let's use it for something useful — like "packing" a string with graphics characters or a Machine Language subroutine. First, let's look at S-80 graphics in general, then "packed string" graphics. Later, we'll "pack" strings with Machine Language.

For graphics, the S-80 screen is divided into a grid of 128 by 48 graphics points, called "pixels". Turning one pixel at a time on with SET or off with RESET can draw pictures, but the process is slooow. For PRINTing characters, the screen is divided into 16 lines of 64 characters. Each of these 1,024 screen positions is numbered 0 to 1023, and may be written to by the command "PRINT @ number". Any character position may have an alphanumeric or "graphics character" PRINTed in it. The "graphics characters" result when the pixels that make up a character position are lit, so a graphics character takes up one full "PRINT @" position, and is two pixels wide and three pixels tall. Each of the possible combinations of the six pixels "on" and "off" is identified by a number. . . from 128 (all pixels "off") to 191 (all pixels "on"). A list of the 64 graphics characters can be seen by RUNning the following "one liner" (you'll have to hit [BREAK] to get out of it):

```
10 CLS: FORI = 128 TO 191 STEP 8:
FORX = 0 TO 7: A$ = STR$(I + X):
PRINT LEFT$(A$, 4); " ";
CHR$(I + X); " "; NEXT X:
IFI = 184 THEN FORA =
0 TO 1 STEP 0: NEXT A:
ELSE PRINT " ": NEXT I
```

The S-80 screen is "memory mapped", which means that each of the character positions is represented in memory by one byte. Whatever is put into one of those memory locations is placed on the screen by the computer's circuitry. We don't actually write on the screen, but writing to the screen memory has the same result, which

gives us an easy way to get the graphics characters onto the screen. We can just POKE them into the screen memory. Huh?? What's a POKE?

POKE and PEEK are BASIC's way of directly altering the contents of a memory location. All we have to know is its address in decimal. Try this: in the S-80 command mode type PRINT PEEK (27648) [ENTER]. This command looks at memory location 27648 and PRINTs its contents on the screen. Now type: POKE 27648, 191: PRINT PEEK (27648) [ENTER]. Look at that!! POKE put a 191 into location 24876. You've just changed the contents of a location in memory. What power!!

Screen memory extends from address 15360 to 16383 (1,024 bytes). Every time we POKE the number of a graphics character into one of those locations, it appears somewhere on the screen. POKE 15360, 191. A block of six pixels will appear in the upper left-hand corner. POKE anywhere you like, but stay on the screen (between memory locations 15360 to 16383). Each line of the display has 64 character positions, so the first line of the display extends from memory location 15360 to 15360 + 63 (remember, the 64 character positions are 0 to 63, not 1 to 64). The second line starts at 15360 + 64, and so on.

Let's see how it works by POKEing a simple picture of the "Enterprise" onto the middle of the screen. Type and RUN:

```
10 CLS
20 POKE 15768, 128: POKE
15769, 128: POKE 15770, 130: POKE
15771, 143: POKE 15772, 143: POKE
15773, 143: POKE 15774, 175: POKE
15775, 159: POKE 15776, 143: POKE
15777, 168: POKE 15778, 188: POKE
15779, 188: POKE 15780, 188: POKE
15781, 188: POKE 15782, 188
30 POKE 15832, 128: POKE
15833, 128: POKE 15834, 136: POKE
15835, 190: POKE 15836, 189: POKE
15837, 188: POKE 15838, 188: POKE
15839, 190: POKE 15840, 189: POKE
15841, 188
```

Oops! That isn't right. The bottom line of the picture is out of place. To fix it, change line 30 to:

```
30 POKE 15836, 128: POKE
15837, 128: POKE 15838, 136: POKE
15839, 190: POKE 15840, 189: POKE
15841, 188: POKE 15842, 188: POKE
15843, 190: POKE 15844, 189: POKE
15845, 188
```

There. . . that's better. We moved the bottom half of the picture four spaces to the right by adding four to each memory location. It's easy to get the memory addresses confused and get a picture that's not straight. POKE graphics are also slow, because BASIC has to interpret each POKE command before it can be executed.

There is a better, faster way to get graphics onto the screen. If we define a string in a program like A\$ = "ABCDEFGHJKLMNOP", and then PRINT A\$, all the letters are PRINTed together, not one letter at a time. The computer stores and PRINTs the entire string as a unit. If we put graphics characters into a string rather than letters, we could PRINT them all at once instead of having to POKE them onto the screen one at a time. Sounds like a good idea, so let's try it:

```
10 A$ = "128,128,130,143,143,143,
175,159,143,168,188,188,188"
20 PRINT @ 448, A$
```

Hmmmm. . . it didn't work. The S-80 thought that we wanted the numbers PRINTed. We can tell it that we want those numbers to represent graphics characters by typing the number in the form CHR\$(X), which stands for CHaRacter \$tring (number X):

```
10 A$ = CHR$(128) + CHR$(128) +
CHR$(130) + CHR$(143) +
CHR$(143) + CHR$(143) +
CHR$(175) + CHR$(159) +
CHR$(143) + CHR$(168) +
CHR$(188) + CHR$(188) +
CHR$(188)
20 PRINT @ 448, A$
```

Now the computer understands that we want graphics, and, since A\$ was

defined before PRINTing, it PRINTs all the graphics characters at once — or nearly at once. Verrrry fast!!

We can “PRINT @” any character position on the screen, but if you PRINT too close to the right edge of the screen, the graphics will “wrap around” to the next line. There are no “end of line” or “edge of screen” boundary markers in the screen memory. It’s just 1,024 consecutive bytes of RAM. The end of the line and screen are functions of the video monitor, not the screen memory.

If we define A\$ to be the graphics characters that make up the first line of the “Enterprise”, and B\$ as the graphics characters that make up the second line (I’ll leave the creation of B\$ to you as an exercise), we can PRINT the whole picture by:

```
30 PRINT@448,A$:
PRINT@448+68,B$
```

Each line of graphics characters gets PRINTed separately, and we have to figure out the offset for the second line so it will line up properly with the first. There is a way to PRINT the whole picture at once, by using three of the cursor control codes. The cursor is the character on the screen that overlies the position where the next character will be PRINTed. The “cursor control codes” — CHR\$(24), CHR\$(26), and CHR\$(27) — don’t PRINT anything, but they MOVE the cursor so that the next character PRINTed is in a different place. CHR\$(24) moves the cursor one space to the left (backspace), CHR\$(26) moves the cursor one line down, and CHR\$(27) moves it one line up. We can PRINT the “Enterprise” by PRINTing the first line, using the control codes to move the cursor one line down and 11 spaces to the left, and then PRINTing the second line. That makes our picture of the “Enterprise” 37 characters long: 15 graphics characters for the first line, 12 control codes to move the cursor down and back, and 10 graphics characters for the second line.

That’s a lot of “CHR\$()+” to type, since you have to type “CHR\$()+” once for each graphics character and control code. A faster method is to put the graphics characters in a DATA statement, then use a FOR/NEXT loop to READ the DATA, put each character into a CHR\$(X), and then add the CHR\$(X) to A\$.

```
10 CLS:A$= “ ”
20 FORI=1TO37: READX:
```

```
A$=A$+CHR$(X):NEXT I
30 DATA 128,128,130,143,143,
143,175,159,143,168,188,188,188,
188,188,26,24,24,24,24,24,24,
24,24,24,24,128,128,136,190,189,
188,188,190,189,188
40 PRINT@448,A$
```

There’s the “Enterprise” again. . . without having to type “CHR\$()+” 37 times. We can PRINT the “Enterprise” anywhere on the screen we want, and it’s fast, about as fast as it’s going to get without Machine Language. Because the computer has A\$ predefined, the only command BASIC has to interpret is PRINT, and that only once.

The last enhancement to the process, the one that turns “print string” graphics into “Super Graphics”, doesn’t speed up the graphics at all. Those DATA statements use a lot of memory. We know that our picture is just 37 bytes long, but the DATA statement to produce it is 135 bytes long. . . not to mention memory used by the FOR/NEXT loop. Each three-digit number in a DATA statement requires four bytes for storage: one byte for each digit, and one byte for the comma that separates the digits.

How can we save memory, and still get the same result? We can store the numbers for each of the graphics characters, 128 - 191, and control codes, 24, 26, and 27, in one byte. We were POKEing graphics characters into screen memory earlier, why not POKE them into a string instead??

Why not, indeed? First, we need a string to POKE into. It has to be 37 characters long, because we need room to put the 37 graphics characters and control codes of our picture. It doesn’t matter what characters are in the string, because they will be replaced by the characters we will POKE in.

```
10 A$= “/////////37/slashes/in/
here//////////”
```

NOW we need VARPTR, since we can’t POKE characters into A\$ until we know where A\$ is in memory. The starting address of A\$ is stored in the memory location pointed to by VARPTR (A\$)+1 (the low order byte in decimal) and VARPTR (A\$)+2 (the high order byte in decimal). To convert the address to decimal (the POKE command requires that addresses be in decimal) we add the low order byte to 256 times the high order byte:

```
20 I=VARPTR(A$): J=PEEK(I+1)
+256*PEEK(I+2)
```

“J” contains the starting address of A\$ in decimal, and the 37 characters (slashes) of A\$ are stored in memory locations “J” to “J +36” (remember that “J” is the FIRST memory location of the string, so the 37 bytes are from “J” to “J +36”, NOT “J +37”). We add line 30, the DATA statement containing the 37 characters that make up the “Enterprise”, and all we have to do is READ the graphics characters from the DATA statement, and POKE them into the string, replacing the characters already there:

```
25 FORK=JTOJ+36: READX:
POKEK,X:NEXTK
```

RUN the program. Type PRINT A\$, and there’s the “Enterprise”.

Now for some REAL magic (well, it seems like magic to me). Since A\$ contains the 37 graphics characters and cursor-control codes that make up the “Enterprise”, we don’t need the DATA statements, VARPTRs, or POKEs any more!!! They were needed to get the graphics into A\$, but now they can be DELETED. DELETE 20-30. Now PRINT A\$. . . there’s the “Enterprise” again. THAT’s the advantage of “Super Graphics”. Once the strings are “packed” with graphics, you don’t need anything except the “packed” strings. What a saving of memory. . . the whole picture is stored in 37 bytes (plus a few more for A, \$, =, and ‘) !!!

Now LIST 10. . . garbage!! It looks like a bad CLOAD. PRINT A\$. . . the “Enterprise” is STILL in there. The line must not be garbage after all, so why does it LIST so badly? The computer uses one byte codes internally and 256 possible decimal values (0 to 255) can be stored in one byte. That seems like a lot. . . until you remember how many codes the S-80 needs. The character set, cursor control codes, graphics codes, and space compression (TAB) codes all need unique numbers. Those 256 values get used up pretty fast.

The Level II BASIC interpreter saves memory when storing programs by assigning each command and operator another one-byte code number (called a “token”). When you type a program line like 10 PRINT A\$, PRINT isn’t stored in memory as the letters P-R-I-N-T.

continued on page 87

PROGRAMMING HINT

"Bats", as presented in the June **SoftSide**, is a program begging for some sound, so I made the following changes and insertions:

```

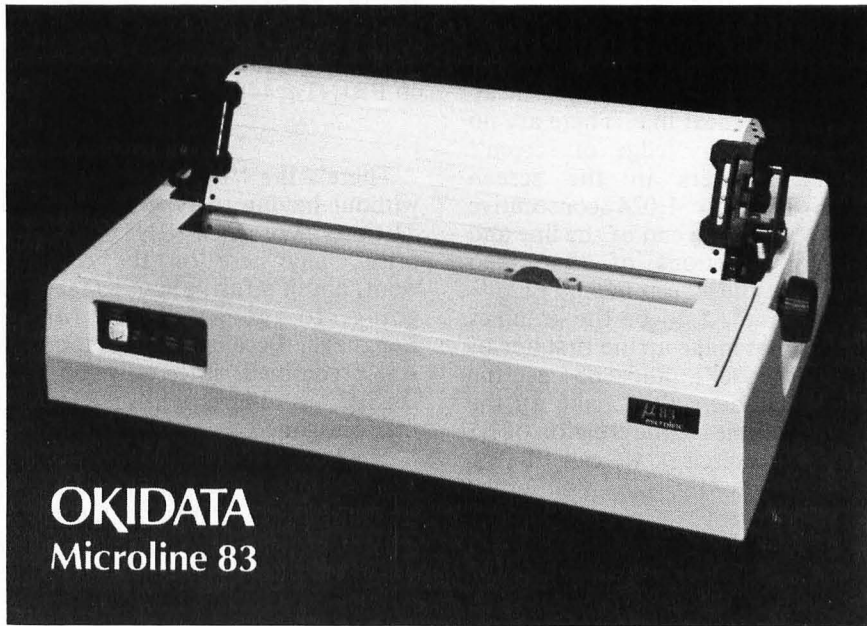
22 DEFINT A-C, E-R, T-Z
80 MS#=STRING$(27,191):S=VARPTR(MS#):SS
=PEEK(S+1)+256*PEEK(S+2):IFSS>32767THEN
SS=SS-65536
85 FORD=SSTOSS+26:READN:POKE,N:NEXTD
90 IF PEEK(16396)=201 THEN POKE 16526,P
EEK(S+1): POKE 16527,PEEK(S+2) ELSE CMD
" T": DEFUSRO=SS: POKE 14308,1
95 DATA 205,127,10,77,68,62,1,105,211,2
55,45,32,253,60,105,211,255,45,32,253,1
3,16,238,175,211,255,201
1520 PRINT @(I#3)+(J#64),">V<"; NT=USR
(7464)
1530 PRINT @(I#3)+(J#64),"YOU"; NT=USR
(8000)
2030 IF Q=72 THEN H=H+1: PRINT@896,"<<H
YPERSPACE>>";X1=RND(19):Y1=RND(11):FO
R D=0 TO 6: NT=USR(7524): NEXT D: GOTO
2200
2300 PRINT @(X1#3)+(Y1#64),"YOU";NT=US
R(8000):A(X1,Y1)=3:GOTO 2400
2310 PRINT @(X1#3)+(Y1#64),"ZAP";FOR D
=0 TO 59: NT=USR(5912): NEXT D:GOTO 400
0
2320 PRINT @(X1#3)+(Y1#64),"BAT";FOR D
=0 TO 9: NT=USR(19300): NEXT D:GOTO 400
0
2600 PRINT @(X(I)#3)+(Y(I)#64),">V<";N
T=USR(7464): A(X(I),Y(I))=2: J=J+1: NT=
USR(7464): GOTO 2700
2620 X(I)=0:K=K+1:N=(RND(7)+22)*256 + R
ND(8)+21: FOR D=0 TO 13: NT=USR(N): NEX
T D:GOTO 2700
2650 PRINT @(X(I)#3)+(Y(I)#64),"BIT";N
T=USR(32000):GOTO 4000
3000 FOR D=0 TO 4: NT=USR(25800): NEXT
D:PRINT @896,CHR$(30):PRINT @896,"YOU W
IN";

```

Now you should be able to hear those bats fry as they hit the fences! Keep up your good work.

Barry Diller
Wynnewood, OH

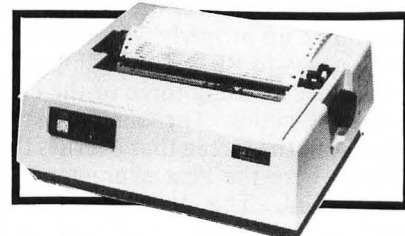
HARDCOPY



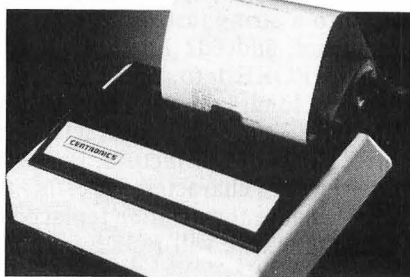
OKIDATA
Microline 83

Direct from TSE HARDSIDE

- NEC 5520 Spin Writer . . (9-5520) \$3195.00
- NEC 5510 SpinWriter . . (9-5510) \$2795.00
- NEC 5530 SpinWriter . . (9-5530) \$2795.00
- CENTRONICS 737 (9-737) \$749.00
- OKIDATA Microline-80 . . . (9-80) \$439.00
- OKIDATA Microline-82 . . . (9-82) \$579.00
- OKIDATA Microline-83 . . . (9-83) \$879.00
- EPSON MX-80 (9-MX80) \$509.00
- EPSON MX-80FT (9-MX80FT) \$619.00
- EPSON MX-100FT (9-MX100FT) \$829.00
- C.ITOH 25cps Daisy . . (9-WP25) \$595.00
- C.ITOH 45cps Daisy . . (9-WP45) \$1995.00



- RS MOD-I Printer
(40-pin) Int. Cable (26-1416) \$59.00
- RS MOD-I & III Printer
(36-pin) Cable (26-1401) \$29.00
- RS MOD-I & III Printer
(40-pin) Cable (26-1415) \$29.00
- RS MOD-I & III LRC Printer Cable
 (9-10) \$29.00
- APPLE Parallel Int. & Cable
(36-pin) (47-936) \$100.00
- APPLE Parallel Int. & Cable
(40-pin) (47-940) \$100.00
- APPLE Asynchronous RS-232C Int
 (47-7710A) \$159.00
- ATARI-Macrotronics Print
 (36-pin) Int(36-936) \$69.95
- ATARI-Macrotronics Print
 (40-pin) Int(36-940) \$69.95



- NEC Tractor-Feed Option
 (9-5000) \$249.00
- BDT Sheet-Feeder
(NEC only) Option (9-5005) \$1495.00
- Microline-80 or 82 Tractor-feed Option
 (9-80-T) \$59.00
- Centronics Zip-Pack Ribbons
(3-pack) (21-01) \$15.95
- SpinWriter Multi-Strike Ribbons
(3) (21-02) \$19.95
- Microline Printer Ribbon
(3-pack) (21-04) \$15.95
- EPSON Printer Ribbons
(2-pack) (21-05) \$29.95
- RS MOD-I Printer
(36-pin) Int. Cable (26-1411) \$59.00
- C.ITOH Tractor Option . . (9-WPT) \$189.00

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

VARPTR Used


continued from page 85

That would use five bytes of memory. No, PRINT is replaced by a one-byte "token", the number 178 decimal (B2 hex), saving four bytes. Humans can't read "tokens", so when a line is LISTed, BASIC converts the "tokens" back to the spelled-out words.

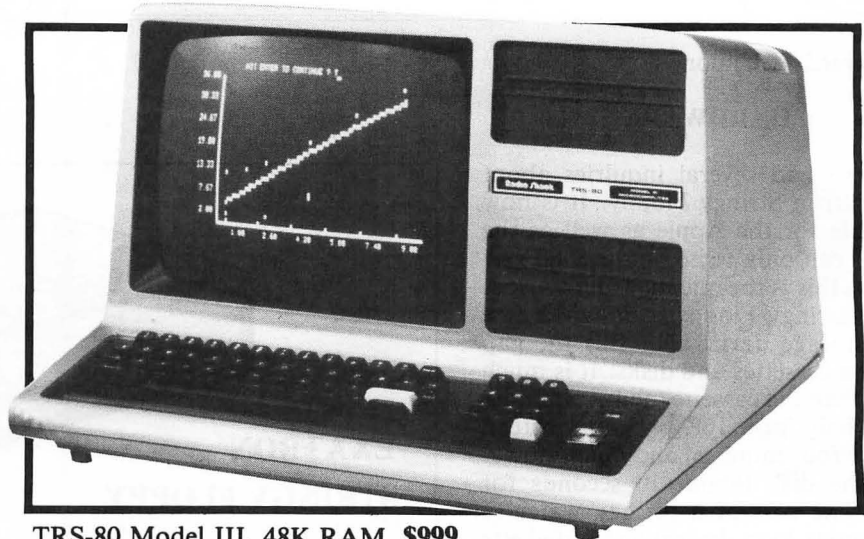
All well and good. . . but there are 125 commands and operators, each of which needs a "token". Since one byte can only have one of 256 values, and most values were in use for graphics and so on. . . they ran out of numbers. Reasoning that graphics characters and TAB codes are meant to be seen on the screen, not in LISTings, and commands and operators will be seen only in LISTings, the geniuses at Microsoft (sincere flattery!!) felt safe in using the SAME numbers for both graphics/TAB codes and command/operator "tokens". Graphics use 128 to 191 decimal and TABs use 192 to 255, while the "tokens" use 127 to 251 decimal. If 191 is in memory, and PRINTed on the screen, all six pixels of one graphics character are lit — our CHR\$(191). But if 191 is in memory, and LISTed, BASIC presumes it's a "token" and expands it to "USING" for the LISTing.

We've put our 37 graphics characters into A\$ where they can be PRINTed on the screen, or LISTed with their program line. When we PRINT A\$, we get graphics. But when we LIST the line with A\$, BASIC thinks the numbers are "tokens" and expands them to their full, readable length for the LISTing. Hah. . . we fooled the machine. But our LISTing is full of TABs, IFs, ENDS, and the other command words. It looks terrible, but it works fine when PRINTed.

A LLISTING of "Super Graphics" cannot be typed into the computer, because it's full of expanded "tokens", when what we want to type in are the graphics codes POKEd into the string. Have your LLISTING contain the dummy strings, the DATA statements, the VARPTR lines, and the POKEs as it looks before the program is RUN the first time. Include a REM statement indicating that the user is to type the program in, RUN it to "pack the strings", then DELETE the lines containing the DATA, VARPTRs, and POKEs. The program, which now contains the "Super Graphics" should be SAVED to disk or CSAVED to cassette.

"Super Graphics" are fast, memory efficient, and, as you've now seen, easy to do. Go draw a picture. 

Hardware from TSE-HARDSIDE



TRS-80 Model III, 48K RAM \$999

Computers

TRS-80 Model III	
16K RAM (#26-1062).....	\$919.00
TRS-80 Model III	
48K RAM (#26-1062 +).....	\$999.00
TRS-80 Model III	
48K, 2 Disk (#26-300).....	\$1995.00
TRS-80 Color Computer	
4K RAM (#26-3001).....	\$359.00
TRS-80 Color Computer	
16K RAM (#26-3001 +).....	\$399.00
TRS-80 Color Computer	
32K RAM (#26-3001 + +).....	\$479.00
TRS-80 Color Computer	
Ext. BASIC (#26-3002).....	\$529.00
TRS-80 Color Computer	
Ext. BASIC 32K	
(#26-3002 +).....	\$599.00

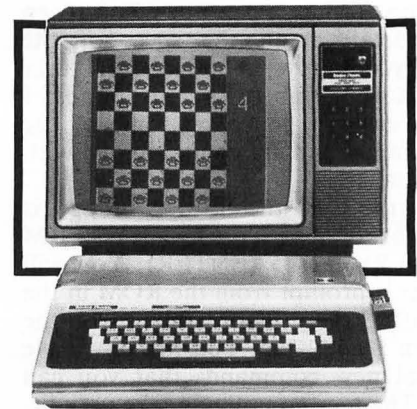
Model I Peripherals

COMM-80 Interface	
(#4-80).....	\$159.00
CHATTERBOX Interface	
(#4-81).....	\$239.00
DISK-80 Interface	
32K RAM (#4-82).....	\$339.00
DISK-80+ Interface	
32K RAM (#4-83).....	\$395.00
LYNX Communications Interface	
(#19-80).....	\$229.00
LYNX Auto-Answer/Auto Dial	
(#19-82).....	\$279.00
RS Exp. Interface	
32K RAM (#26-1140-32).....	\$389.00
16K Memory Kit	
TRS-80 Interface (#5-1102).....	\$39.00
ORCHESTRA 80	
(#15-03).....	\$79.95
Dual Joysticks	
for Color Computer (#26-3008)...	\$24.95
CTR-80A Cassette Recorder	
with cable (#26-1206).....	\$59.95
DIGI-TALKER MOD I	
(#4-DT1).....	\$189.00
DIGI-TALKER MOD III	
(#4-DT2).....	\$189.00
80-Graphic Board	
(#26-80G).....	\$139.00

Model I Disk Drives

HARDSIDE 40-Track Drive	
(#7-40).....	\$299.00
HARDSIDE 80-Track Drive	
(#7-80).....	\$449.00
PERCOM Data Separator	
(#7-03).....	\$29.95
PERCOM Doubler-II	
(#7-07).....	\$149.95
HARDSIDE Extender Cable	
(#7-02).....	\$15.95
HARDSIDE 2-Drive Cable	
(#7-04).....	\$29.00
HARDSIDE 4-Drive Cable	
(#7-05).....	\$39.00

TRS-80 Color Computer 32K RAM \$599



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

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



by Edward E. Umlor

HARDWARE

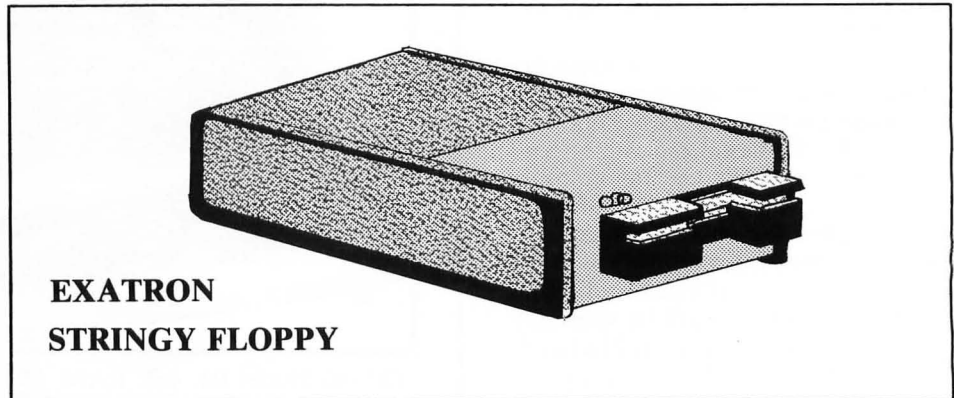
I have had several inquiries about the Exatron Stringy Floppy. It is now available for the Apple as well as the S-80. I can only write on the S-80 version, as this is the one that I have used.

The Stringy Floppy is designed as a mass storage device that fills the gap between cassettes and disks. It is much faster than the cassette (about a minute and a half for a 16K program instead of over four minutes) and much slower than the disk (about 10 seconds for 16K). The cost of a Stringy wafer (a continuous loop device) and a diskette is about the same, so the cassette is still the least expensive way to go. Money is speed and efficiency in this here data game. Stringy Floppies can be more than one-unit systems and are patterned more after disks than cassettes.

There are several similarities between the disk and Stringy Floppy in handling Assembly Language programs. You save the program using the first byte location, last byte location, and start location format. When the program is loaded, it will auto start without having to use the “/” and “ENTER”. BASIC programs are saved in a manner similar to the cassette format. I am not going to go into detail about the specs of the Stringy Floppy. The manufacturer does a good job of that in its ads. I will, however, touch on some of the commands and its compatibility with other devices.

On the S-80 the Stringy Floppy is activated by typing SYSTEM (ENTER) /12345 (ENTER). You will receive a prompt from the ROM in the first (0) Stringy Floppy telling you the system is activated. You will now have several new commands that you can run. The new commands take very little memory as the symbols are all that is placed in memory. The subroutines are contained in the ROM located in the first Stringy Floppy.

@NEW is the command that is used to test, format, and verify a new wafer (the “@” is the symbol that triggers the new Stringy Floppy ROM commands). The @ command must be run on every new wafer as it sets the BOT (beginning of tape) and the EOT (end of tape) markers as well as running a media check. Many wafers will not fully verify, but will still save programs without loss of data.



@SAVE# (# = number of program on wafer 1, 2, etc.) is used to save a BASIC program on a wafer.

@LOAD# is used to load a BASIC/Assembly Language program from the wafer.

All the commands have a multiple drive format and you will receive several useful utilities with the package. I would like to address myself to the compatibility aspect of the Exatron Stringy Floppy. The Stringy Floppy hooks into the S-80 on the bus expansion port. Yes, you can run one directly off the keyboard!! Some of you might not be aware of the fact that the front connector on the left side of the Expansion Interface is a bus expansion port. The Stringy Floppy is less expensive than an E.I. and disk drives, but is still fully compatible when you do add these goodies. You would then be able to select which medium of storage you wanted — cassette, Stringy Floppy, or disk. I have used ours with all three methods hooked up and operational, and could transfer files between all three.

Apple owners should have this same freedom. Since each option plugs into its own output port, I would expect the same overall freedom of data transferal to be applicable. I would like to get one to check out on the Apple, but will have to wait for time and funds to become available. Larry Baker of Kansas, I hope this has been of some help to you.

SOFTWARE

This month's software review will probably seem a little brief to some people. The old saying — When you have seen one disassembler, you have seen them all — will apply somewhat to

this piece of software. It is the first utility for the S-80 color computer to pass across my computer. I am speaking of the new offering from Interpro in Manchester, NH. The “Color Computer Disassembler” is now in production for all you Assembly Language persons out there. I know you BASIC types could care less what is present in the ROMs, but there are those who cannot wait to tear into the 680 BASIC interpreter.

Well, the day has arrived for you assembly types that want to be able to call up ROM routines. The format of the display/printout from this disassembler is based on the tried and proven #1-address of byte, #2-hex code of address, #3-6809 mnemonic for the byte, and #4-relative jump address if any. The documentation is short, but sufficient to get you going in good shape. Interpro even warns you that the first 32 of the ROM in the color computer is a table of addresses and will not decode properly. The program is well prompted for the beginners out there who are just starting to get into the catacombs of the Assembly Language. I am just getting started myself on the Z-80 set.

The only conclusion that I can make is, this utility is a must for the Assembly Language person with a color computer. I don't know of too many people (even if they stay in BASIC all the time), who own S-80 Mod Is, that don't have at least one Z-80 disassembler in their library. This is the first one that I have seen for the 6809.

Well, I guess that about wraps up another session of fuss'n'muss in the world of newdom. See y'all in September.



Hardware Corner

by Edward E. Umlor

Here I is again! It's that time to bend the old eyeballs again. I have started to receive some mail and I will try to start answering it.

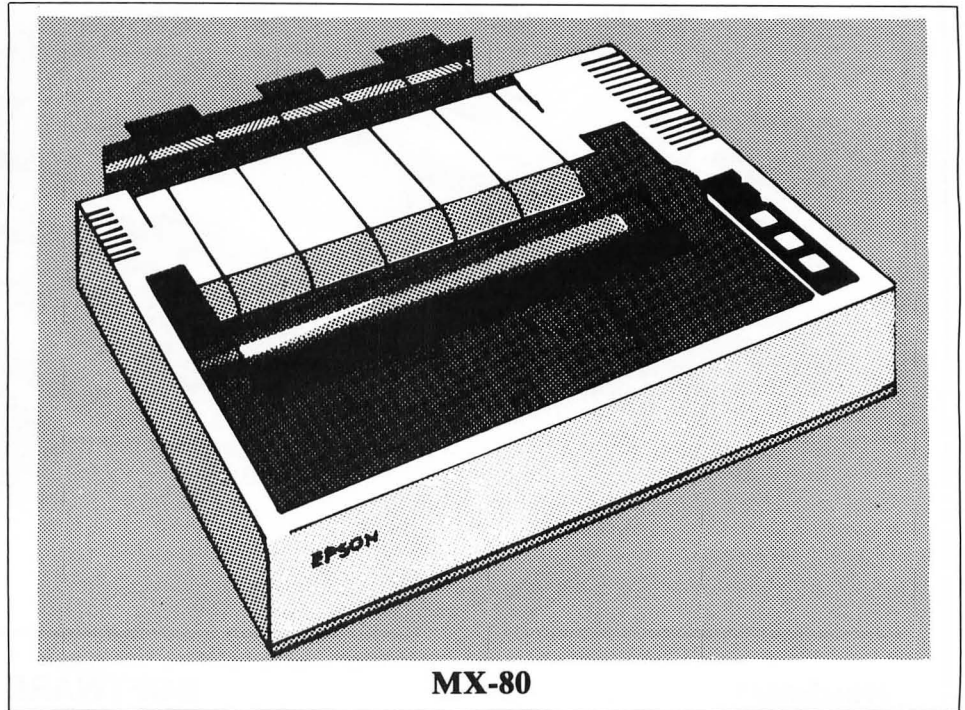
The Apple and MX-80 were the subjects of a couple of letters compliments of Mr. Frank Greene of Hedgesville, WV. The system that he wrote about is an Apple II with a TYMAC parallel interface and an MX-80 printer. The system displays 40 characters per line on the video and also prints 40 characters per line. To overcome the 40 character limitation on the printer, strike a CTRL-I (which turns off the monitor screen) and 80 N (which sets 80 characters per line). Another CTRL-I will turn the video back on. There is one little problem, however. To quote the letter — "By the way, the use of these commands will elicit a raspberry duet and error message from the Apple and printer, but such bad manners shouldn't be taken personally. Everything works fine after they quiet down." This letter was soon followed by another one which said that his Apple had learned some manners. The use of CHR\$(9);80;CHR\$(78) will accomplish the same results without the backtalk and CHR\$(9) will restore the video.

I believe that in an article on programming hints for printers, the use of CHR\$(9) was mentioned. Here it is not so important to set up strings that are equal to your control functions, although this same method could be used in a program for data output.

While on the subject of the MX series of printers, there are several things I should pass along at this time.

The MX-70 is a seven-bit word printer and works well as is with the Macrotronics cable for the Atari. However, this is not so with the several cables for the S-80 Interface. Many cables sold as the 26-1401 will require that pins 31 through 36 be isolated (electrical tape does well, so does removing the pins from the connector). There is a reset line in these top pins that is grounded in the interface. The low level means reset.

The MX-80 is an eight-bit word printer and works well with the S-80 Interface. If you want to make full use of its capabilities for underscoring,



MX-80

etc., you will have to modify the standard cable again. Pin 14 of the 36-pin connector is held at ground by the interface. This line allows some computers to select auto/non-auto line feed. Leave the dip switches in the printer the same as delivered. Isolate the wire for pin 14 (it should be the eighth wire from the high-numbered pin side of the cable). Most cables have 34-conductor ribbon cable. Using a sharp knife, separate the eighth wire from the seventh and ninth. The cut should be about two to three inches long. Lift the wire up and cut it in the middle of the loop that is formed. Strip and tin about one-eighth inch on each exposed end and solder in a single-pole single-throw (SPST) switch. Shack part number 275-624 @ \$1.59 works well in this application. When the switch is in the OFF position, the software will have to furnish the linefeeds, and in the ON position works just like a standard S-80 printer. If you are going to use the Macrotronics cable for the Atari, check pin 9 of the 36-pin connector for ground. This is the eighth bit line and it must be grounded. You can tell that as soon as you try to print something. If you can only get the graphics characters, then pin 9 is not grounded. You will not be able to use the graphics on the Atari using the

Macrotronics cable as the graphics are in the upper 128 codes and you will be able to produce only the lower 128 codes.

Just one more thing about the MX-80 and I will sign off. Epson has now furnished us with the parts to fix a problem called FUZZY PRINT. This usually shows up the worst in the condensed double-strike mode. Here are the codes to set up the mode. Your line of text to print should be at least 130 characters long: LPRINTCHR\$(15);:LPRINTCHR\$(27) + "G". The 15 is condensed mode and the ESC G is for double-strike. If your printer does not strike cleanly over each line (you should print a minimum of one full page), contact your nearest Epson warranty service center or Hardside for the fix. There is no charge for parts or labor, so don't hesitate to have it done. The fix also shortens the BEEP from three seconds to about one second maximum. You can obtain your nearest service center's location from Epson at (213) 378-2220, and say HI!! to Laura from me.

Well, that is about all for now from the old GRANITE KNOGGIN for this time out. It sure is nice to know there are other keyboard pounders out there. Keep pounding and HAPPY COMPUTING.

Connect with ATARI



Hardware

ATARI 400 Computer System, 16K RAM	\$339.00 (#36-401)
ATARI 400 Computer System, 32K RAM	\$519.00 (#36-402)
ATARI 800 Computer System, 16K RAM	\$829.00 (#36-800)
ATARI 800 Computer System, 32K RAM	\$929.00 (#36-801)
ATARI 800 Computer System, 48K RAM	\$999.00 (#36-802)
ATARI 410 Program Recorder	\$69.00 (#36-803)
ATARI 810 Disk Drive	\$499.00 (#36-810)
ATARI 822 Thermal Printer	\$389.00 (#36-820)
ATARI 825 Printer (80-col)	\$769.00 (#36-825)
ATARI 830 Acoustic Modem	\$179.00 (#36-850)
ATARI 850 Interface	\$179.00 (#36-855)
16K RAM Module for the ATARI	\$99.00 (#36-854)
32K RAM Module for the ATARI	\$169.00 (#36-855)
MACROTRONICS Printer Interface (36-pin)	\$69.95 (#36-936)
MACROTRONICS Printer Interface (40-pin)	\$69.95 (#36-940)
ATARI Joystick Controllers	\$19.95 (#36-3005)
ATARI Paddle Controllers	\$19.95 (#36-3004)
ATARI CX-70 Light Pen	\$74.95 (#36-70)
Dust Cover for ATARI 400	\$7.95 (#16-40)
Dust Cover for ATARI 800	\$7.95 (#16-03)

ROM programs

Basketball	\$29.95 (#36-BASK)
Chess	\$34.95 (#36-CHS)
Editor/Assembler	\$49.95 (#36-ASE)
Music Composer	\$49.95 (#36-MUSE)
Star Raiders	\$39.95 (#36-STRDS)
Super Breakout	\$34.95 (#36-SUPB)
Tele-Link	\$24.95 (#36-TEL)
3D Tic-Tac-Toe	\$29.95 (#36-3TTT)
Video Easel	\$29.95 (#36-VIDEO)

SOFTWARE on Disk

VISICALC from Personal Software	\$199.95 (#36-VICL)
MAILING LIST	\$24.95 (#36-279002D)

SOFTWARE on Cassette

Star Trek 3.5	\$14.95 (#36-200025T)
Deflection/Simon Says	\$9.95 (#36-200078T)
Mountain Shoot	\$9.95 (#36-200079T)
Angle Worm/Croton Diversion	\$9.95 (#36-200092T)
Fleet	\$6.95 (#36-27701BC)
Connect Four	\$6.95 (#36-277020C)
Letters	\$6.95 (#36-277022C)
Reverse	\$6.95 (#36-277023C)
Zap	\$6.95 (#36-277024C)
Lander	\$6.95 (#36-277019C)
Chase	\$6.95 (#36-277017C)
Computer Acquire	\$20.00 (#36-237002C)
Conflict 2500	\$15.00 (#36-237001C)
Lords of Karma	\$20.00 (#36-237001C)
Biorhythms	\$14.95 (#36-BIOR)
Blackjack	\$14.95 (#36-BLACK)
Hangman	\$14.95 (#36-HANG)
Kingdom	\$14.95 (#36-KING)
3-Dimensional Graphics Package	\$29.95 (#36-3D-G)

TERMS: Prices and specifications are subject to change. HARDSIDE accepts VISA & MASTERCARD, Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). 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.



SARGON II

At HAYDEN, The Best Has Gotten Better.

Sargon, the program that came in first in the Creative Computing Microcomputer Chess Tournament, has become Sargon II. The game has been vastly improved and now has a faster response time. A new Level 0 has been incorporated for beginners. The board is easier to pre-set and there is now a Hint mode that provides suggestions from the computer. Sargon II took on the maxi-computers in the West Coast tournament and finished in the money! Shows more thinking power than you ever expected.

SARGON II SPECIAL! 20% OFF

Special

Sargon II 16K Level II Cassette \$29.95 (S-80)	\$23.95
Sargon II 24K Cassette Machine Language \$29.95 (Apple)	\$23.95
Sargon II 32K Disk \$34.95 (S-80)	\$28.95
Sargon II 48K Disk Machine Language \$34.95 (Apple)	\$28.95

*offer expires September 15, 1981

**SoftSide
selections**
6 South Street, Milford NH 03055
For Orders Only 603 673 0585





APPL-L-ISP from Datasoft, Inc.

Yet another "big machine" language has found its way into the micro market. "APP-L-ISP" is the first implementation of Lisp for the Apple computer, and after several hours of experimentation I consider it to be a very successful adaptation.

The package consists of three items: the program diskette, the "APP-L-ISP" user's manual, and the book, **LISP**, by Patrick Winston and Berthold K. P. Horn, both of M.I.T. Unlike many other language packages you might have used in the past, new users of "APP-L-ISP" are expected to have at least some exposure to standard Lisp. If you aren't familiar with Lisp, the text will provide some help. The only possible difficulty is that the book describes MacLisp, the dialect of Lisp developed at M.I.T., and "APP-L-ISP" does have some significant differences. Due to these differences, learning "APP-L-ISP" will not be as easy as learning BASIC.

The program diskette contains several programs. One of these is "APP-L-ISP" itself. This is the environment in which the Lisp functions are created and executed. "APP-L-ISP" is loaded upon booting the system. Because Lisp is an interpretive language like BASIC, this can be thought of like running disk BASIC.

Another program on the diskette is the Lisp editor. I've used several mainframe Lisp editors and the "APP-L-ISP" editor was surprisingly easy to use while still maintaining the true flavor of a Lisp editor. In fact, many of the commands are identical to mainframe Lisp editors. After just 15 minutes I was completely at ease with the editor.

The editor is loaded into "APP-L-ISP" by issuing the command:

(LOAD 'EDITOR)

Editing is performed on the property list of a function. To edit a function, the user edits the property list obtained with the GETD function. Thus, to edit a function named TEST, the user would type:

(EDIT (GETD 'TEST))

after LOADING the editor from disk.

The diskette also contains a MacLisp emulator which is obtained with the command:

(LOAD 'MACLISP)

This emulator permits the use of the functions: DELETE, MAPCAR, REVERSE, DEFUN, APXMND, REMPROP, DEFPROP, APPLY, GET, NCONC, FUNCALL, and SUBST.

The Lisp itself is a somewhat stripped down version of MacLisp with added features applicable to the Apple machine. The standard functions such as CAR, CDR, and CONS are present, but other functions you may have grown used to using, such as NULL, LESSP, and ZEROP, must be defined by the user. Fortunately the user is supplied with sufficient rudimentary functions to make these definitions simple.

"APP-L-ISP" also includes a trace package to help debug functions. The system also allows the user to direct all I/O to a printer to obtain hard copy records of functions and their operation. The user also has full access to the Apple's graphics capabilities. In addition to graphics the system features:

- 16-bit integer arithmetic including multiplication, division, and the MOD function.

- PEEK, POKE, and CALL functions which allow access of up to 650 machine code routines.

- A random-number generator.

- PDL function for game paddle input.

- Debugging facilities including BREAK and BAKTRACE.

As well as those elements already discussed, the diskette includes a pretty-printing facility in addition to two sample programs: "HANOI" (The Towers of Hanoi) and "DOCTOR" (described in Winton and Horn's text).

In all, it's an impressive package with valuable applications in artificial intelligence research in a microcomputer environment. For \$124.95 you can teach yourself and your Apple a new language and actually enjoy the experience.

Mark A. Ohland

ATARI 3-DIMENSIONAL COMPUTER GRAPHICS PACKAGE

from Sebree's Computing

One of the newest and most rapidly expanding fields in programming is the realm of computer graphics —

specifically three-dimensional graphics. Very few people have the hardware and software experience to write a good 3-D graphics program. If you've ever tried to do so, you can appreciate the knowledge required to write one of any real significance. The complexity of the mathematics and programming far exceeds the average computer owner's level of training. (Unless you're a genius and/or have taken a college course in Advanced Calculus 3000, and even then it's still confusing.)

This three-dimensional graphics package by Tim Hays is just such a program (significant, that is, not confusing). The overall design is a good one, although there are some serious drawbacks. Its manual, though a little hard to read, is detailed enough for the average person, and whether you are experienced or just want to give it a try, there are several experimental programs in the back that teach you (and your Atari) some new tricks.

Along with the main graphics data base program (called USER.3D), there are four demonstration programs included. The sample 3-D objects provided include simple drawings of a pyramid and the space shuttle. And of course, you can always type in your own.

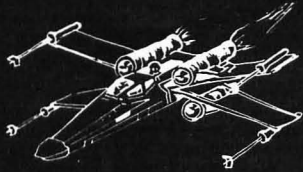
When entering the data (known as a shape table) into the data base for your drawing you are asked for the X, Y, and Z coordinates of two points — the first point (or intersection) to plot, and the next point to which the previous point will be connected by a straight line. Using this method you can "build" a three-dimensional shape out of the lines to form a kind of "stick drawing" which you can then rotate and/or view from a different location or angle.

The coordinates X, Y, and Z correspond to the horizontal and vertical position of the point and its DISTANCE from you. This means that a line (two points) can be drawn "pointing" straight at you, and if viewed from head on (X=0, Y=0, Z=0) it will appear as only ONE point because the other point is DIRECTLY behind the first. However if you view both points from an eastern angle (X=10, Y=0, Z=10), they will then appear as a normal horizontal line because you are now looking at it from

continued on page 94

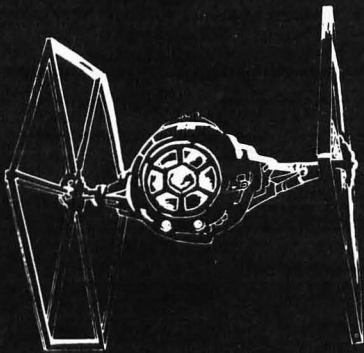
X-WING

II by
Chris Freund



For the thousands who have enjoyed X-Wing Fighter, X-Wing II presents a totally new element in the game!

You are the pilot of an X-Wing fighter . . . Your Mission, Destroy the Death Star!



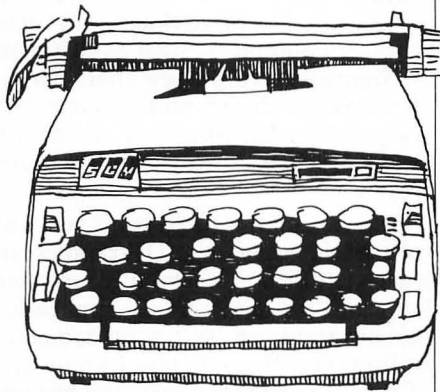
Where X-Wing I left Death Star looming on the screen, X-Wing II lets you guide your fighter into the trench, find the exhaust port, aim and fire — all the while avoiding enemy fighters. Excellent graphics, 12 levels of play, and extensive INKEY\$ commands make this one of our most exciting "real time" games.

S-80 16K Cassette . . . \$9.95



Typing Tutor

by Roy Groth



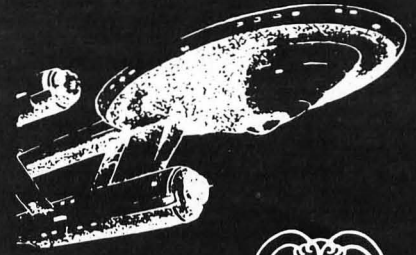
Wish you were a better typist, but don't want to take (or pay for) a class? Teach yourself to type with the aid of your micro-computer. With TYPING TUTOR you will be quizzed and graded, but you set the pace at which you learn. TYPING TUTOR is a set of programs that lets you become as good a typist as 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-80/16K/Cassette \$15.95



SoftSide August 1981



STAR TREK III.5

ALL NEW VERSION!

by Lance Micklus

Now with Sound Capability and Increased Speed of Execution.

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 in the region. Klingon battle cruisers are also present, so the utmost care is needed.

Star Trek III.5 includes: playboard 8 by 3 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.

S-80 16K Cassette \$14.95

S-80 32K Disk \$14.95



REVIEWS

continued from page 92

a PERPENDICULAR perspective (i.e. from the side).


You can rotate shapes to any angle including distance, because if you vary the distance of an object, you are really varying the ANGULAR DIAMETER (i.e. the apparent size of the object based on the distance it is from you).

As I mentioned before, there is a selection of tricks for your Atari included in the back of the manual. The first is a way of allowing you to get the text window on TOP of the screen instead of the bottom. The others include more colors in Hi-Res (Graphics 8), mixing the different graphics modes on the screen, and even increasing execution speed WITHOUT any hardware modifications.

One of the drawbacks is that the program is written in BASIC, and is therefore slow at times even with the speed-up trick used. A good example of this is evident in the AUTO.3D program when the computer is rotating four drawings of the space shuttle at once. Another drawback is that there is no way of saving your data base to tape or disk. The instructions tell you to write them up in data statements and

add them to the program, but this is quite simply a poor idea and an easy way out. There is no reason why a data LOAD/SAVE program could not have been added to the program package. The way it is now, you have to create a separate copy of the program containing your data statements for that particular drawing. That is, unless you like typing in data over and over again or you're just playing around. If you make a mistake after it's typed in, you have to go back and retype the whole thing. Also, this particular 3-D package lacks some of the nicer options available in packages for other types of computers, such as distorting the figure; creating a new character set; or drawing a design on the screen with a joystick or light-pen and saving it to tape or disk using multi-colored shapes.

This is a decent program if you want to experiment with 3-D graphics and learn a little more about your Atari, but for a serious graphics application, this program just isn't good enough. It is, however, the only one I know of as of now, but as more software becomes available for the Atari, I think we will see a more complete system designed for the serious user.

Alan J. Zett 

Welcome to . . .

THE RACES



THEY'RE OFF!

Eight horses surge down the track, straining for the lead, with your horse struggling in the pack.

They round the turn and head into the stretch. Your horse shoots from behind, catching the lead horse. They cross the finish line.

The Win, Place, and Show horse results are printed on the screen, along with each bettor's race winnings and total daily winnings.

You collect your winnings and decide if you want just to watch, or bet on the next race. You study the odds, place your bets, and select the track speed—fast (dry), average, or slow (wet).

The horses are at the starting gate, jumping and snorting. You raise the gate, and the next race is underway.

Each horse gallops forward randomly. Spectators squirm and shout as they urge their horses to win.

You have all the track action and thrills. Plenty of winners—and losers! Now you can use your computer to find out what it takes to win at the track. Good Luck!

Requires 16K Send check, or charge it to Visa S-80 Tape—\$9.95 or MC. (Print charge number and S-80 Disk—\$14.95 expiration date—Phone 313-627-2877 for charge if you wish)

WE GIVE IMMEDIATE SERVICE!

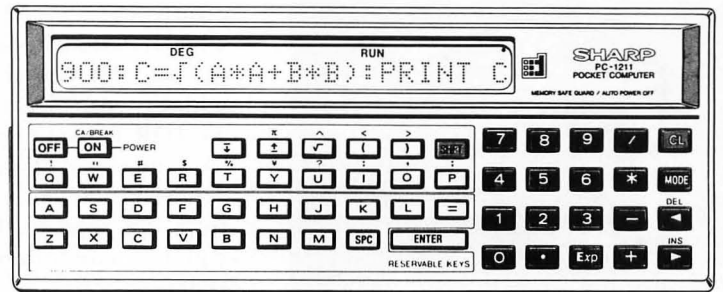
ECHO PRODUCTS INC., 335 MILL, ORTONVILLE MICH. 48462

DEPT. S Dealer Inquires Invited



PC-1211

Pocket Computer



For Engineers, Scientists, Businessmen

USE IT WHEREVER YOU GO — IN THE FIELD, CLASS, COMMUTING.

Handy pocket size: 6 oz., 2 3/4 x 6 7/8 x 11/16"

BASIC Language: Most of the features of the Level I TRS-80 microcomputer. Program capacity of 1424 steps.

Editing Functions: Cursor shifting, insertion, deletion, line up and down.

Prerecorded Programs: A growing library of programs is available on cassette tape. Memory retains data and programs even with the power off.

Includes: Hard case, long-life batteries, Instruction and Application manuals, BASIC textbook, two templates, and 90-day Sharp warranty.

Special Introductory Price **\$249.00** (includes shipping).

BRAND NEW! CE-122 Printer/Cassette Interface

Use to load prerecorded software, store and save programs. Use to print out programs and calculation records.

Size: 11 3/32 x 3 3/4 x 1 3/8" (same dimensions with PC-1211 inserted)

Printer: 16-digit, 1 line/second, standard paper

Compatibility: completely compatible with TRS-80 Pocket Computer.

Includes: Ni-Cd rechargeable battery, AC adapter, carrying case, and cassette cable.

Price: **\$150.00** (includes shipping)

Combination Price: (PC-1211 plus CE-122) **\$359.50** (includes shipping)

To order

Payment by Personal check, Master Charge, or VISA #. Massachusetts residents include 5% Sales Tax.

or

for additional information: **ATLANTIC NORTHEAST MARKETING, INC.**

P.O. Box 921 Dept. SS61
Marblehead, MA 01945
617-639-0285

Let our fingers do the typing!

With a **SoftSide** Disk or Cassette subscription you can get each month's TRS-80, Apple, or ATARI programs delivered on disk or cassette with every issue of your **SoftSide**. No more hours of typing. No hunting for typing mistakes. The programs for your computer are tested and ready to go the day you receive them.

If you already receive **SoftSide** magazine, you will receive credit for the remainder of your subscription toward your new cassette or disk subscription.

Magazine \$24/year
 Magazine and Cassette \$75/year
 Magazine and Disk \$125/year

Make your **SoftSide** library complete with back issues of **SoftSide Apple Edition**, **SoftSide: S-80 Edition**, and **PROG/80**. Programs, games, complete documentation and lots more! If you have missed out on any past issues, now is the time to order.

SoftSide: S-80, and **SoftSide: Apple** back issues (magazine only) \$2.50

PROG/80 back issues \$4.00

Magazine with Cassette programs \$9.95

Magazine with Disk programs \$14.95

New Super **SoftSide** back issues (Magazine Only)

August or September, 1980 \$3.00

October, 1980 to present \$3.50

For descriptions of the contents of our back issues, please see the May, 1981, issue of **SoftSide**.

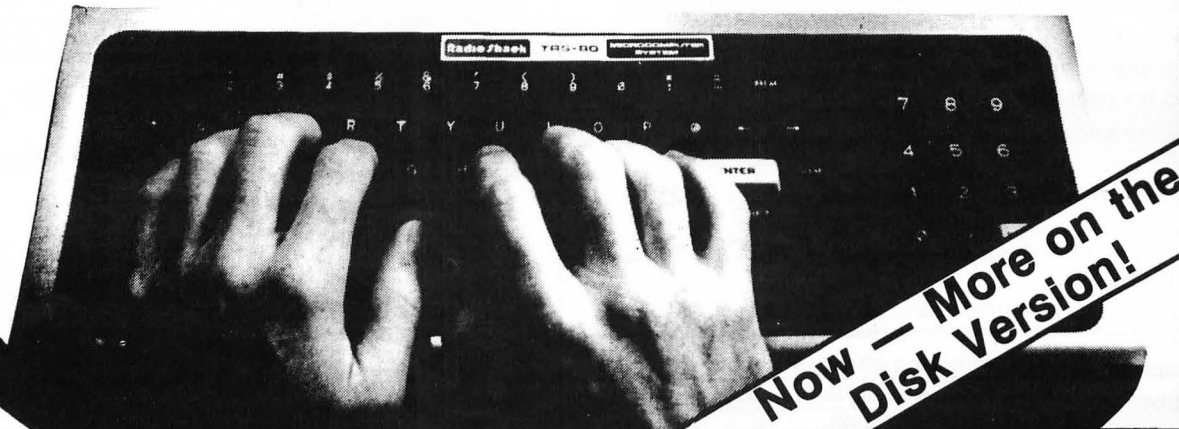
**SoftSide
 Selections**
 6 South Street Milford NH 03055
 For Orders Only 603-673-0585

SoftSide
 cassette & diskette

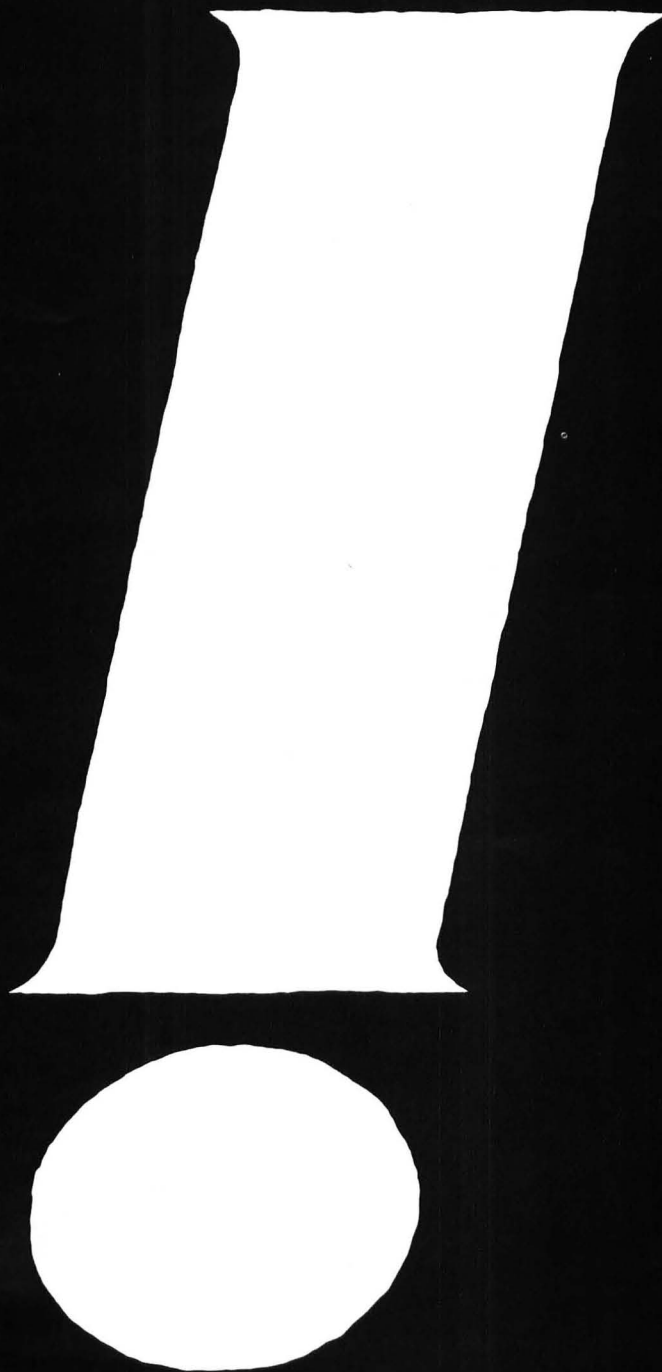


Collectors! Protect your **SoftSide** back issues, Volumes I and II, or any publication of your choice, with these durable wood grain vinyl binders with inside pocket and clear spine sleeve for easy identification. Holds and protects 12 back issues. A regular \$4.95 value, SALE priced at \$3.95. Free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

Small	\$3.95
8½ X 11	\$7.95

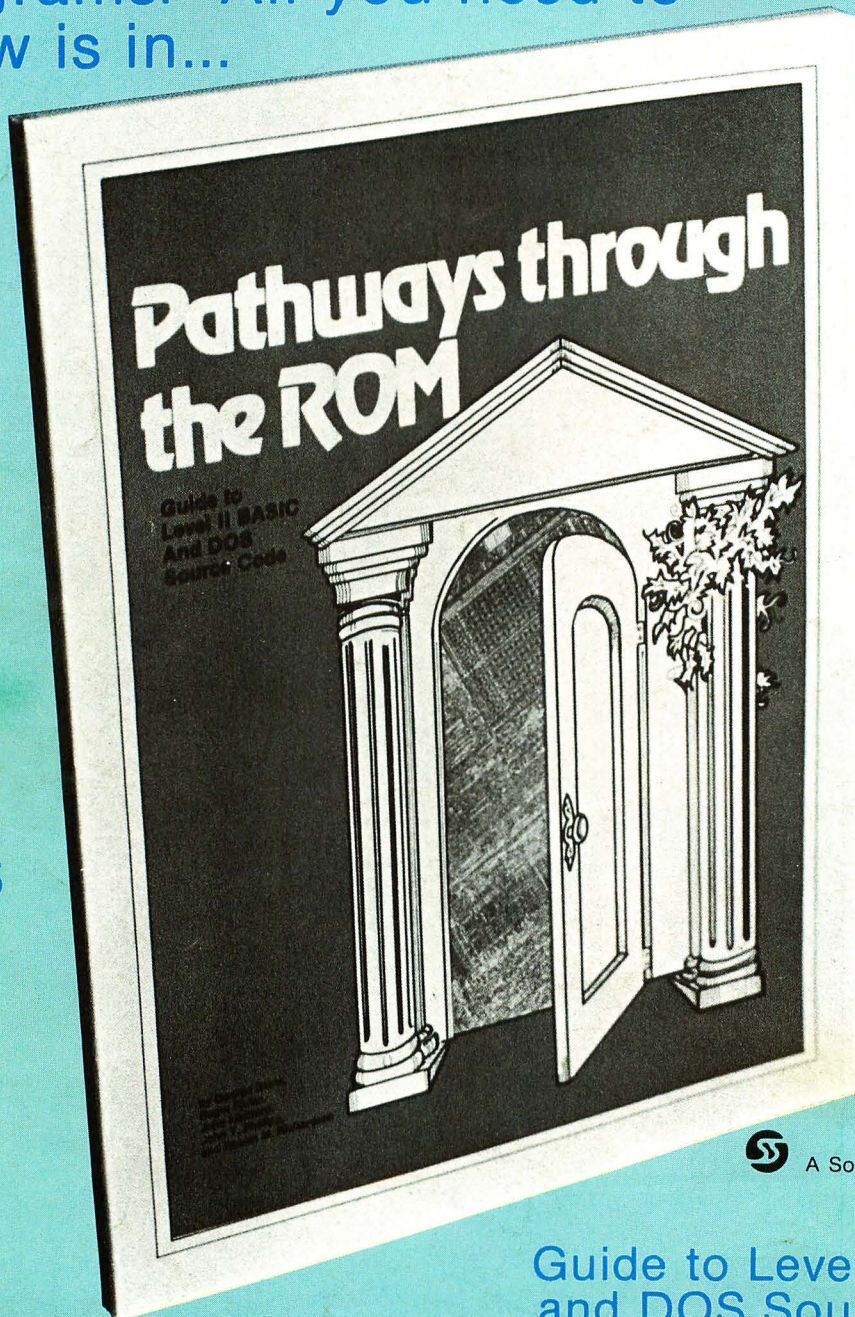


Entertainment Software will
never be the same....



!-\$TRING... a revolutionary approach

Unlock the hidden power of your computer for fast and easy programming! Use ROM routines in your BASIC and Assembly Language programs! All you need to know is in...



ALL
ONLY
\$19.95
plus \$1 shipping

INCLUDES:

SUPERMAP
From Fuller-Software (\$18.95)

**TRS-80
DISASSEMBLED
HANDBOOK**
by Robert Richardson (\$10.00)

HEX MEM
by John Phillip
Monitor written in BASIC

**Z-80
DISASSEMBLER**
by George Blank



A SoftSide Publication

Guide to Level II BASIC and DOS Source Code

Description of the contents of the Level II BASIC ROM by memory locations, by function, and in lesson format. Includes several BASIC and Assembly Language programs in listing format to examine and use ROM routines.

