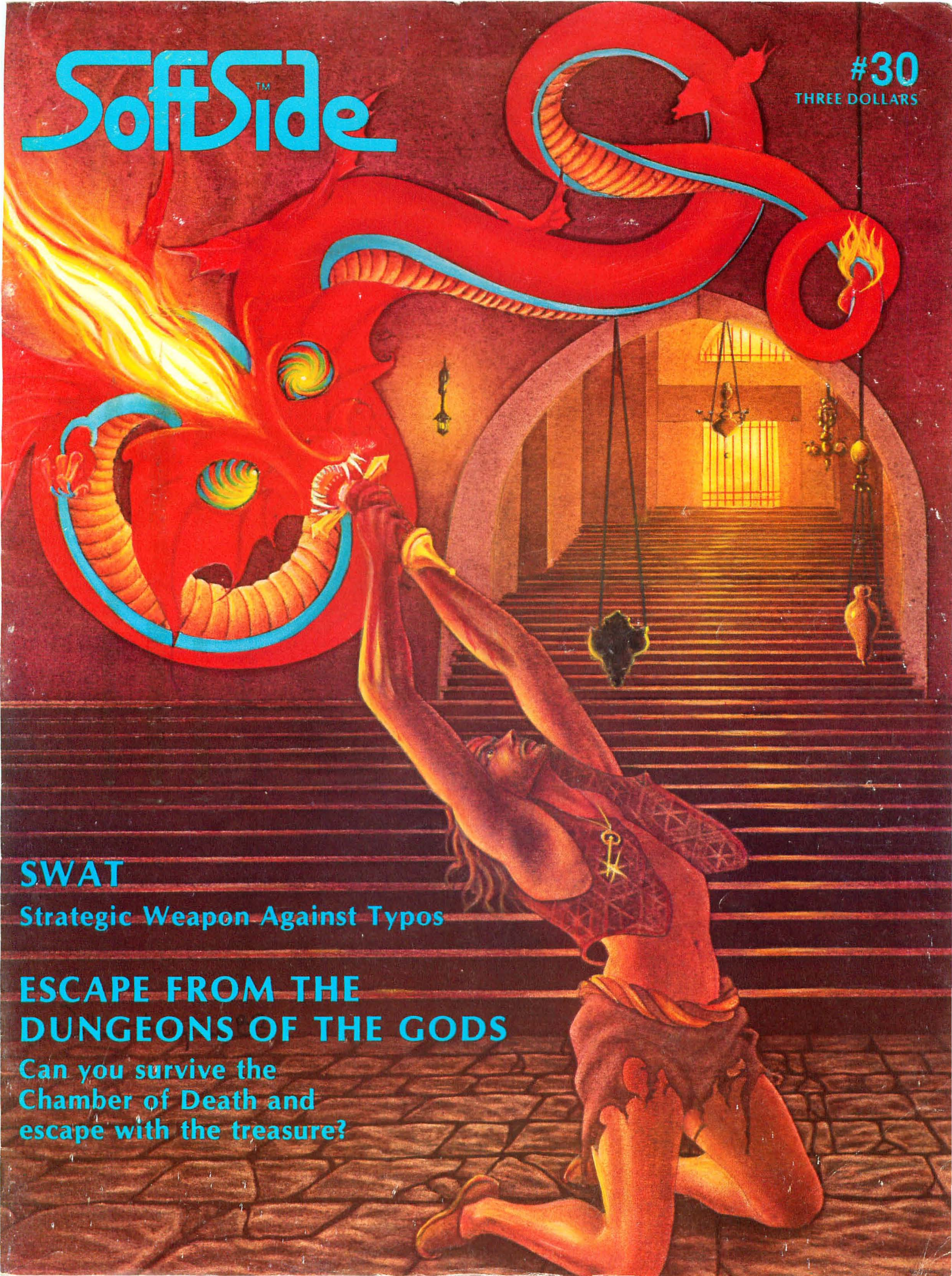
SoftSide™

#30
THREE DOLLARS



SWAT

Strategic Weapon Against Typos

ESCAPE FROM THE DUNGEONS OF THE GODS

Can you survive the
Chamber of Death and
escape with the treasure?

STOP TYPING!

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

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

Cassette Version (CV)

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

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

Disk Version (DV)

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

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

To order, use the card provided in this issue.

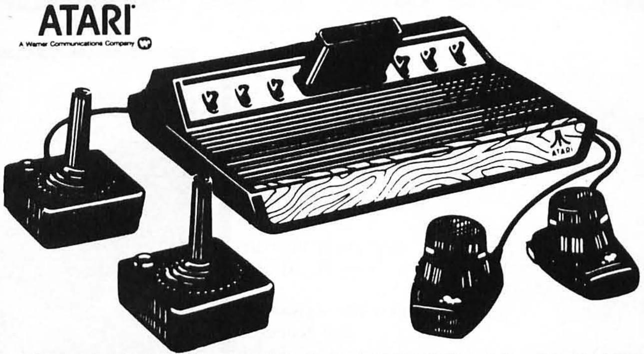
VIDEO COMPUTER SYSTEM

● Create excitement with your home TV. Sharpen mental and physical coordination with a full library of challenging, sophisticated video games. ● Bright, crisp color (on color TV). ● Realistic sound effects (through the TV speaker). ● Interchangeable controllers and cartridges to enjoy the full range of **Video Computer System** games. ● Difficulty option switches so the games get harder as you get better. ● Special circuits to protect TV. ● System comes with Combat Cartridge, Joystick and Paddle Controller, AC adaptor and TV switch box. ● Similar to coin operated games. ● Used with player's own TV set.

SUGGESTED RETAIL $199.95

ATARI
A Warner Communications Company

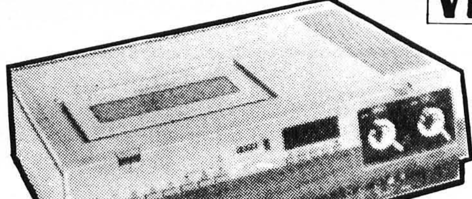
OUR PRICE
$179.99



- Asteroids
- Warlords
- Breakout
- Superman
- Home Run
- Bowling
- Canyon Bomber
- Backgammon
- Championship Soccer
- Casino
- Space Invaders
- Video Pinball
- Adventure
- Night Driver
- Circus Atari
- Maze Craze
- Football
- Video Checkers
- Dodge 'Em
- Slot Racers
- Missile Command
- Othello
- Basketball
- Game Cartridge Storage Unit
- Indy 500
- Video Chess
- Video Olympics
- Air-Sea Battle
- Tic-Tac-Toe
- Golf

2/4/6 HOUR VIDEO TAPE

VHS™



OUR PRICE
809.99

SUGGESTED RETAIL $899.95

● New 6-hour "Super Long Play" video tape recorder with soft touch controls. ● Switchable to "Long Play" 4 hour and Standard 2 hour recording. ● Compatible with other 2/4 and 2/4/6 hour VHS systems. ● Record programs off the air or record home movies with an optional TV camera. ● Automatic tape threading. ● Hot pressed ferrite video heads. ● Direct drive video head cylinder motor. ● Azimuth recording system. ● Built-in UHF and VHF tuner allows recording one station while viewing another. ● Tracking control. ● 100/ solid state integrated circuitry. ● Pause mode; complete with remote pause control. ● Auto-digital memory counter provides automatic stop while rewinding. ● Complete with all accessories for hookup to all types of TV sets and antennas.

INTELLIVISION

● The video television system which provides an endless variety of game play and involvement. ● Compatible with any color TV set. ● Utilizes a full selection of Mattel game and learning cartridges. ● Each cartridge comes with overlays which fit directly over the two individual hand controllers for easy game play decisions (no additional controllers). ● Includes Las Vegas Poker/Blackjack cartridge. ● Objects of the game screen can be maneuvered in 16 different directions by the object control disc. ● 16-bit microprocessor provides a full range of sound effects, music, color and an extraordinary high level of resolution.

SUGGESTED RETAIL $299.95

OUR PRICE
$279.99

TV not included.

MATTEL ELECTRONICS

- Astrosmash
- Space Armada
- NASL Soccer
- Auto Racing
- Adventure
- Boxing
- NHL Hockey
- Horse Racing
- Space Battle
- NFL Football
- Tennis
- Las Vegas Roulette
- Sea Battle
- NBA Basketball
- PGA Golf
- Backgammon
- Armor Battle
- Major League Baseball
- U.S. Team Skiing
- Checkers

TSE 14 South St., Milford, N.H. 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

master charge THE INTERBANK CARD
VISA®
AMERICAN EXPRESS

SoftSide™

PUBLISHER
John G. Grow

ASSOCIATE PUBLISHER
Nancy Lapointe

EDITOR
Randal L. Kottwitz

ASSOCIATE EDITOR
Guy S. Allred

SOFTWARE MANAGER
Bill Kubeck

PROGRAMMING EDITOR
Jon Voskuil

EDITORIAL DEPARTMENT
Rich Bouchard
Belinda Hutchins
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Alan J. Zett

ADVERTISING MANAGER
Sue Rowland

ART PRODUCTION MANAGER
Lynn Wood

PRODUCTION DEPARTMENT
Lynda Fedas
Denise Lafleur

CUSTOMER SERVICE
Cindy Schalk

DEALER SALES
Kathie Maloof

STAFF
Mary Edwards
Donna Jean
Steve Justus
Bea Kimball
Doris Miller

Photographs by Dean F. H. Macy

SoftSide Vol. 5, No.9

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

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

If you have any problems or questions with your subscription please write to *SoftSide* Publications, 515 Abbot Drive, Broomall, PA, 19008

Cover illustration by Bill Giese

FRONT RUNNERS

FEATURES

CONTENTS

DEPARTMENTS

APPLE™/SIDE

Articles

Reviews

ATARI®/SIDE

TRS-80®/SIDE

Information Inundation and the LaserDisc

by Randal L. Kottwitz

It's everywhere! It's everywhere! Every day more and more information cascades into our lives. Each of us is in the process of becoming a large, walking database, ready for access from any number of "on-line" inputs and outputs. Take a moment to survey the environment in which you sit reading this magazine. How many inputs are just sitting there, openly grasping for your attention or waiting for the slightest motion on your part to pour a tidal wave of information your way — the telephone, the television, your computer, magazines, books, newspapers, advertising fliers, ad infinitum. It seems that everyone is fighting to receive our attention, regardless of our desire to give it. The typical American is approached and asked to spend money an average of 752 times per day. More than one computerist, familiar with the jargon, has asked how soon additional RAM modules will be available for the human brain. Obviously, the time is long past due for us to take a firmer grasp on the information available to us and find more efficient methods for its organization and utilization.

Enter — The Videodisc. The more accurate phrase would be, "Enter into the consumer marketplace — The Videodisc." Industrial versions of the interactive videodisc technology have been in use since 1978. However, the consumer market is only now on the verge of seeing prices drop to an affordable level. As with any piece of information technology, the major lag in its penetration of our everyday life will be the inevitable problem of software development.

I'll not attempt a detailed report on the current state-of-the-art. The available information could easily fill a complete issue of ***SoftSide***. However, here are a few definitions and specifications to focus this discussion: The videodisc system format to which I'm referring is the LaserVision format. The less expensive CED (Capacitance Electronic Disc) and VHD (Video High Density) formats currently have substantially limited applications due to their reduced random access capabilities. Information is stored on the LaserVision discs in the form of pits on an injection-molded, aluminum plated core encased in a protective, transparent plastic disc. The bits of information encoded in the pits are read by reflecting a low-powered laser off the reflective core. One side of a disc has a capacity of 54,000 screens of still-frame or 30 minutes of continuous video programming. (Current manufacturing limitations decrease the controllable program area by approximately ten percent.) That translates into an available digital storage capacity of 10^{11} bits of information.

Currently, laserdiscs are limited to Read Only capabilities. However, one Japanese company has unveiled a working prototype of a laserdisc recorder and the development of such technology seems inevitable. Due to the physical "burning" of the pits by a slightly more powerful laser, laserdisc "writing" will be permanent in nature. However, the staggering amount of storage available on a single disc and the high-speed random access capabilities of laser technology will considerably mute the impact of this limitation.

The pricetag for adding the necessary hardware (currently available for the Apple and ATARI® computers) to your system can run from $900 to several thousand dollars. The price of handler software on magnetic media varies according to its complexity. What little laserdisc software is available seems to be settling in the $20 to $30 range. All of these figures will be subject to fluctuations as the new industry finds its niche in the marketplace and the technology costs can be spread over the masses.

The substantial advantage which positions the laserdisc so firmly in our future is its ability to integrate more forms of information than ever before. Microcomputers brought their own brand of information control to the video screen in their manipulation of digital information. The major tool they've previously lacked has been the ability to control the presentation and handling of information in a non-digital (analog) format. The laserdisc delivers the best of both worlds into the hands of the microcomputer by offering common storage for digital and analog signals. Now, the micro has the capability to reach onto one storage medium, spinning at 1800 RPM, move the information which it's been looking for to the magnetic diskette **and** to control the flow and presentation of analog video/audio signal between the videodisc and the monitor.

For example: A single videodisc could carry a multi-volume cooking course presenting three major categories of information. In the first category, portions of the storage area could be devoted to live, video, action demonstrations of cooking techniques to be watched at full speed with sound, or in single frames or slow motion without sound. The dual soundtrack could carry voice-over descriptions of the demonstrations on one channel and related information on spice selection, chef's tricks, etc. on the other. Secondly, other areas of the storage space on the disc could carry digital program data for weights and measures conversion programs, diet customization programs and other software to be loaded into the RAM of your computer. The third area, still frame video information controlled by and integrated with microcomputer software, has fewer models to emulate. A quiz on the cooking techniques could be administered by the microcomputer with wrong answers automatically cuing review frames of demonstrations on other parts of the disc. Still frames of efficient kitchen designs could be called to the screen, digitized by the computer, and then modified by you to fit your existing floorplan. If there were a portion of one of the demonstrations you had a hard time understanding, you might be able to stop it at that point, ask for further information and cue up a more detailed explanation/presentation of the information elsewhere on the disc, complete with a question/answer session to assure that you understood the information before returning you to the more general demonstration.

You may very well think I've got my head in the clouds and am telling you about a technology which will be available in five or ten years. In truth, *Master Cooking*, an interactive laserdisc with chef Pierre Franey and food critic Craig Claibourne, is currently available and utilizes many of the techniques described above. It is not designed for control by a microcomputer and therefore does not incorporate the computer software capabilities. However, all of the features I've mentioned are possible now and are waiting only for someone to utilize them and bring them to market.

There is no doubt that the laserdisc is a major step forward, bringing the information age to our doorsteps. It will have a powerful impact on almost every kind of information we now use in our daily lives and will give us access to information we never dreamed useful before simply because it was too difficult to bring to practical application. The question must be asked: "Which brings more to the handshake of computer and laserdisc — the microcomputer, with its controlling capabilities or the laserdisc with its storage capabilities?" The answer to that question will determine which of the two appliances will become part and parcel of the other in the future. Sony® has announced production of a laserdisc player utilizing a Z-80 microprocessor with 1K of RAM. How soon will that RAM become expandable to a point that the resident controlling microprocessor on the laserdisc player starts to take on attributes of the microcomputer as we now know it? Indeed, the principal information appliance in the home of the future may well be known as "laserdisc" rather than "computer."

How would you like ....

A FREE DISK OR CASSETTE SUBSCRIPTION?

Each month **SoftSide** publishes a translation of a program from a previous issue. Since the **Translation of the Month** has been so well received, we're offering an even greater incentive than ever before to those of you who put the necessary effort into producing a good translation for one of the other computer systems we support.

A one-year subscription to the Disk Version of **SoftSide**, or an eighteen-month subscription to the Cassette Version is the incentive. That's a value of $125 for the disk subscription, or $112.50 for the cassette subscription; a handsome reward for a winning translation.

What do we look for in a translation contest winner? Here are some of the most important qualifications:

■ Your entry should be a translation of one of the main programs from a past issue. **K-Byters, One-Liners**, and other short programs don't qualify for this contest. (We will, however, consider translations of shorter programs in their own right, especially if they contain unique features or enhancements.) In general, we are looking for translations of programs which are a CHALLENGE to translate. Some of the listings we publish are written in more or less "generic" BASIC, which can be typed into another computer with few changes. Although these require the least effort to translate, they are also the least likely prospects for Translation Contest winners. We'd much rather see translations which require considerable creativity and ingenuity to rewrite and adapt.

■ Your translation should be thoroughly tested and completely bug-free. Just converting program lines doesn't automatically ensure a workable translation. Be sure to use-test your translation as carefully as you would test a program you had written entirely from scratch.

■ Your translation should take advantage of the unique features of the computer for which it is written. The objective of a translation is not to simply duplicate the operation of the original program in a mechanical way. Rather, the translation should be written in such a way that it "fits" the capabilities and conventions of its host computer. This is especially true of programs which use graphics, and should be kept in mind for such minor things as keyboard layout as well (use of such special keys as arrows, ESC, CTRL, CLEAR, etc.). Also be careful with screen formatting; a word that spills over into the next line, because of a PRINT statement that wasn't properly rewritten, betrays such carelessness that we'll probably reject your translation automatically.

■ Your entry should incorporate any improvements and enhancements that you can add to make it an even better program. Don't feel that you have to limit yourself to the boundaries of the original. (On the other hand, don't go overboard and destroy the character of the original by completely rewriting it!) An enhanced translation is much more likely to catch our attention than a line-for-line duplicate, and it will have more value to our readers.

It's not necessary to include the kind of extensive documentation with your translation that we require of original programs. If most of the originally published documentation applies to your translation, simply say so. You should, however, for the benefit and interest of other programmers, include descriptions and explanations of any changes or enhancements you've made.

All Translation Contest entries must be submitted on disk or tape, with documentation in printed or typed form. Media will be returned only if accompanied by a self-addressed, stamped envelope.

Send to:
SoftSide Translation Contest
6 South Street
Milford, NH 03055

INPUT

From our readers

Dear *SoftSide,*

I would like to comment on your magazine. It would have to be one of the best. I had not heard of *SoftSide* until 2 weeks ago. My school had an Apple II+ donated to them earlier this year. Right away, I knew I was interested in microcomputer programming. Ever since then I have been writing and programming my own programs in BASIC. I am only thirteen but have been sincerely considering being a professional microcomputer programmer. Your magazine has helped me make up my mind. I would like to start a cassette subscription, but I think it is educational programming in your programs, looking at the break up of each program and learning how each piece works. I also think it is a good idea to have a magazine like this so you can learn and have fun at the same time. I would like to thank you for helping me make my decision and encouraging computer programming for people of all ages.

A lot of people in our school think of computers mainly for games, games, games. I sincerely regret this and am hoping that they come to realize that computers are excellent for anything and everything. I think this is the greatest thing that has come into my programming life. Thank You.

Brian Vrshek
Mt. Prospect, IL

Dear *SoftSide,*

I am just writing to tell you what a good magazine you publish. I think that *SoftSide* is the best computer magazine I've ever seen, but there is one complaint I have about your magazine, the problem is that you blame all of the errors in your programs on the printer, but the fact is that a good printer will not drop lines or leave out parts of a line. In *SoftSide* you publish good quality programs, but some of them don't work after typing them, I'm not blaming it all on you guys at *SoftSide*, because I'm not a very good typist and I do make a lot of typing errors, but after I go over the programs three or four times for typos, the programs still don't work. Some of the programs that don't work are: *Convoy, Kidnapped, Quest 1,* and *Titan. Convoy* doesn't work at all, *Quest 1* keeps me in room 0 for the whole game, and *Titan* has an error in the buying and selling routine.

Although I have this complaint it will not stop me from getting another years subscription when this one runs out because no other magazine offers good programs, articles, and advertisements, at least not as good as yours.

Now that I'm through complaining I will give you a complement or two. I own an ATARI® and there aren't very many magazines that offer programs or games for ATARI® users, in fact you are just about the only one that does. I have noticed that a lot of the people writing in are complaining about your format (combined format) I personally like it better this way, because I can look at and compare other BASICS to ATARI®'s. I think that it is up to you how you make your magazine.

Walter Lavelle
San Jose, CA

Editor's Reply: Take a look at the *SWAT* program in this issue — we at *SoftSide* think the use of this utility on every program we publish will really help find errors. As for the bugs, we try to kill as many as possible before they slip into the magazine. Some continue to slip by, but we do our best. We publish a "bug report" in the next possible issue. Have patience, the bugs are few and far between.

Dear *SoftSide*,

In reviewing the Radio Shack Acoustic Modem I, Lance Micklus accurately described the inadequacies of this piece of substandard hardware, but did not offer a solution to those already owning this device.

My neighbor, once a die-hard TRS-80® owner, was in the same boat. Having invested a bundle in this system, he could neither afford to rectify his mistake nor divulge to his spouse that his hardware wasn't exactly "up to snuff" — most wives think computers are less than a necessity anyway!

To overcome his problem, we had the phone company install a handset for the hearing impaired. This unit has a small thumbwheel in the handle and can be used to increase the volume. Since then, he has had no problems — so, if your modem is a little deaf and your checkbook a little light, call "Ma Bell."

I was also amused with Lance's saga on Route 17 in northern New Jersey. It's a shame he didn't know about SOFTWARE CITY in River Edge, approximately two miles from the Radio Shack Computer Store and COMPUTER UNIVERSE. They have one of the largest inventories of software for the TRS-80®, ATARI®(which I own), Apple, PET and VIC computers I've discovered so far. And, they're discounters who backup what they sell! Had he visited them, his ego would not have been bruised — they probably have everything he offers (no, I don't own stock in the store).

Anyway, I hope my little tip on "hearing-impaired modems" will be of help. Keep up the good work...Your magazine is pretty good (It would be great if your bug-control were just a little better.)

Paul Pettennude
Wayne, NJ

Editor's Reply: Thanks for sending in the information I'm sure it will be of help to some of our *SoftSide* readers.

ANSWERS TO QUESTIONS SUBMITTED IN INPUT

From:
Thomas Bank II
Mechanicsburg, PA

Question:
"If I were to send you a program by any chance would you put it on the DV and not in the magazine? (I am not a DV subscriber.) Also, your ad states that for a translation you would give me a $100 Software Certificate if you use the translation. If I sent you my translation of a program, you used it, and I got the certificate, could I use the certificate and $25 out of my pocket to get the DV subscription?"

Answer:
Yes, your translation could possibily be used for the DV program but it might also appear in the magazine. As for the question about the Software Certificate, yes, you can apply it to a DV subscription, in fact in this issue you will see a new ad in reference to translations. We are encouraging translations of programs in the back issues of *SoftSide* and instead of the Software Certificate we are offering a free, one-year subscription to *SoftSide* DV, a $125.00 value or an eighteen month CV subscription valued at $112.50. For you who are already subscribers this will extend your subscription .

From:
Michael Mahaffey
Manhattan, KS

Question:
I purchased your November '81 DV. I tried it in my Model III TRS-80® without success. I tried it on a friend's Model III and it still would not boot up. However, it works very well on a Model I. I have examined your advertisement for the DV and nowhere is there any mention that the DV is only for the Model I. I suggest you either mention the fact the DV is Model I only, change the DV to be readable by the Model I and the Model III, or put out a separate DV for the Model III.

If there is anything I can do to get the DV to boot on my Model III please tell me. Otherwise, I am going to have to get a Model I friend to read the disk and copy it to a disk I can read.

I have enjoyed your magazine very much. I have copied some of the programs onto my machine and modified them. I would like to ask one thing. WHY AREN'T THE LINE NUMBERS EVENLY NUMBERED!!! I can't imagine anyone these days that doesn't have some kind of renumbering program. How much trouble could it be to renumber a program before submitting it?? How much trouble could it be for you to REQUIRE submitted programs be evenly numbered. It makes it TEN TIMES EASIER to turn on auto line numbering when typing in a program from a magazine and just worry about typing the program statements!! You can't type in a program with the auto on and change line numbers as you go because the line number that you changed will probably be jumped to from somewhere else in the program. PLEASE, PLEASE!! publish programs numbered evenly!

Answer:
Michael, you and many other Model III DV Subscribers will be happy to know that as of the May '82 issue we are making available both a Model I and a Model III Version of *SoftSide* TRS-80® DV. All TRS-80® DV subscribers will be notified of this in the May disk mailing. Subsequent issues will be sent on Model III Disk to those who request this.

In reference to your question about evenly numbered lines, programs submitted to us with evenly numbered lines are published that way. In our author's guide we encourage authors to use even line numbers but not all of them do. For many reasons we do not alter their programs. The cost of staff time required to renumber all the programs published each month would be phenomenal and the chances of creating errors in the process would also be likely. Thanks for the input. It is always appreciated.

K-Byters

ANOTHER PROGRAMMING CHALLENGE

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

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

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

Here then are the official rules:

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

Send submissions to:

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

HINTS & ENHANCEMENTS

From our readers

ATARI® MICROTEXT

ATARI® *Microtext* is great! The April issue of ***SoftSide*** came in the mail a few days ago and I just got through keying in the third and last (I hope you reconsider) part of this mini-wordprocessor. After working with the program a short while I thought of a few brief enhancements which I'd like to pass along to the other readers.

The first has to do with alerting you when you are approaching the dimension limits of T$ — this is the string that holds all the text. Line 120 shows T$(14000) — I have a 32K system with a disk drive so I've set mine to 11000. Try the following:

```
ADD TO THESE LINES
130 SETCOLOR 2,0,12:SETCOLOR
1,0,4:SETCOLOR 4,0,12
4000 SETCOLOR 4,0,12
ADD THIS LINE
6010 IF LEN(T$) >10900 THEN
SETCOLOR 4,3,4
```

What this does is set the background and border of the TV screen to white and the characters to black, which is much easier to see (also see ***SoftSide***, February 82', page 68-POKE YOUR ATARI®). Then every time a line of text is added T$ is checked. If it exceeds 10900, the border surrounding the screen turns red. This gives you time to finish your thought and save the text to tape or disk. When you have saved the text the screen returns to normal.

The second has to do with saving a little bit of memory. Lines 4075-4090 & 5075-5090 are identical. What I've done is combine them into a short subroutine which can be called from either section of code. A little more can be saved if you don't care for the tape or disk option (Lines 4000 & 5000). By going with the disk only, the save and load sections of code would look like this:

```
4000 GOSUB [newly written subroutine]
4100 OPEN #2,8,0,F$
4210 - 4400 [remain the same]
5000 GOSUB [newly written subroutine]
5100 OPEN #2,4,0,F$
5210 - 5400 [remain the same]
```

That's it! Now get busy and send in some other enhancements.

Al Chilcott
Greensboro, NC

ATARI® OUTER SPACE ATTACK

Sheldon Leemon's *Outer Space Attack* for the ATARI® in the March issue has certainly delighted the nine-year old gaming freak in our household (and the adults also); however, he probably didn't count on klutzes such as myself who manage to send their player off the screen into a program crash. Consequently I added an error trap in line 90 which fixes it and keeps me sailing:

```
90 HP = HP + 4*(STICK(0) = 7)-4*
(STICK(0) = 11):TRAP 90: POKE
53248,HP:TRAP 40000
```

I look forward to each new issue of ***SoftSide*** with keen anticipation and especially thank you for the ***SoftSide** Database* (a real god-send!).

Edsel Mikkola
Minneapolis, MN

TAB FOR THE ATARI®

For those of you who have been frustrated by ATARI®'s lack of a TAB command, here's an idea. You probably already know that POKEing location 85 will place the cursor at the specified column, but that doesn't help when you are trying to format output to a printer. I have discovered a simple means to simulate a tab function which works with both the screen and printer. It only takes one line of code and translates easily.

```
10 DIM TAB$(40):TAB$ = " (40 spaces) "
```

Thereafter, Tab functions can be called as in the following example:

```
20 FOR X = 1 TO 30 STEP 5
30 PRINT TAB$(1,X);"This is TAB";X
40 NEXT X
```

Note that LPRINT can be substituted for PRINT in line 30.

I hope your readers can put this idea to good use.

Jim Alley
Interlochen, MI

WORD WRAP-AROUND

Here is a subroutine for the Apple II that will automatically prevent word wrap-around on the Apple's 40-column screen. This is useful when, as is often done, one desires to print a long message in one PRINT statement.

In order to use the subroutine, set the variable KK$ to the unformatted string, then call the subroutine. An example would be:

```
600 KK$ = "THE RAIN IN SPAIN FALL
     S MOSTLY IN THE PLAIN."
610  GOSUB 50000
620  END
```

The subroutine may, of course, be renumbered to fit any particular program. Here it is:

```
49999  REM  PRINT FORMAT SUBRTN
50000 PT = 1
50010 C = 0
50020 WD$ = ""
50030 X$ =  MID$ (KK$,PT,1):PT =
     PT + 1: IF PT >  LEN (KK$) GOTO
     50110
50040  IF X$ <  > " " THEN WD$ =
     WD$ + X$: GOTO 50030
50050  IF C +  LEN (WD$) = 40 THEN
      PRINT WD$;: GOTO 50010
50060  IF C +  LEN (WD$) = 39 THEN
      PRINT WD$: GOTO 50010
50070 WD$ = WD$ + " "
50080 C = C +  LEN (WD$): IF C >
     40 THEN  PRINT :C =  LEN (WD
     $)
50090  PRINT WD$;
50100  GOTO 50020
50110  IF C +  LEN (WD$) > 40 THEN
      PRINT
50120  PRINT WD$; RIGHT$ (KK$,1):
      RETURN
```

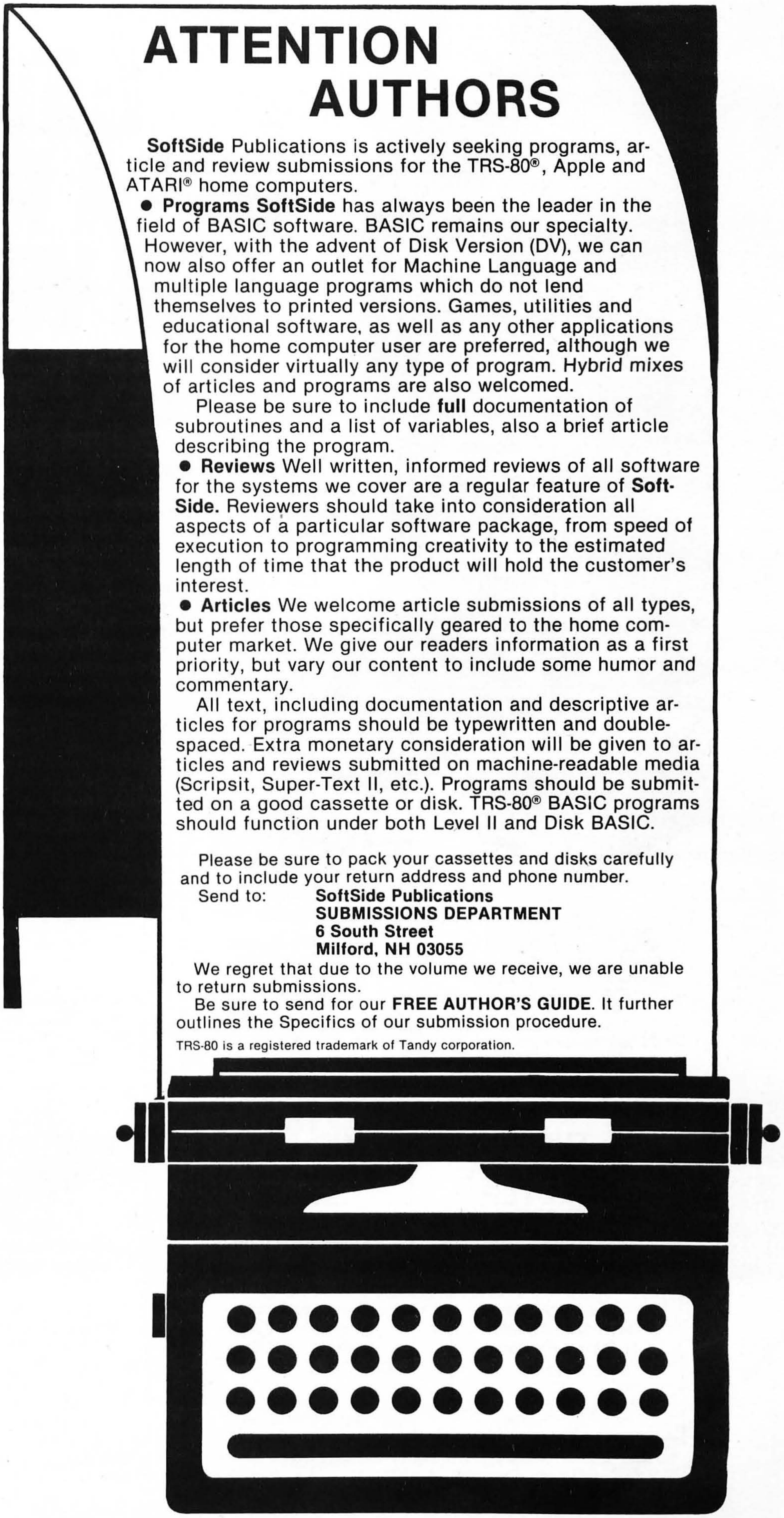
Fred Condo
Covina, CA

TRS-80® MAZE SWEEP

The following changes will update and save to disk the top ten scores for the TRS-80® program, *Maze Sweep* (February, 1982). Lines 30 and 32 are changed by the routine at lines 2200-2300, so lines 30-35 must have exactly 21 spaces between each set of quotation marks. (There are also exactly 21 spaces between the quotes in line 2165.) This allows room to POKE the name and high score directly into program memory, where those strings are stored. This method is easily adaptable to other similar games, and could be adapted for use with cassette systems as well.

```
20 DIMB,B1,B2,FC,FL,I,I$,J,K,M,N$,P,Q,SC
(11),SC$(11),T,Z,M(13),P(200),VP(11),P1(
11)
30 SC$(10)="Tim Pierce    7140":SC$(9
)="Rik Pierce         7090":SC$(8)="Timoth
y Pierce   6850":SC$(7)="James Garon
  6640":SC$(6)="Arnie Gross        6530":S
C$(5)="Rik                   6380"
32 SC$(4)="Rik Pierce        6360":SC$(3)
="Tim Pierce        5930":SC$(2)="Willie
           5740":SC$(1)="Rik The Great
 5180"
35 SC$(11)="MAZSWEEP             "
37 FOR I=1TO11:VP(I)=VARPTR(SC$(I)):P1(I
)=PEEK(VP(I)+1)+PEEK(VP(I)+2)*256:SC(I)=
VAL(RIGHT$(SC$(I),4)):NEXT
400 PRINT@18,SC$(10);:GOSUB9000:FORI=1TO
26
2060 SC$="":PRINT@474,CHR$(206);:PRINT@4
76,;:LININPUTSC$:IFSC$=""THEN2060
2063 CLS:FORI=2TOLEN(SC$):'Alters to low
er case
2064 IFMID$(SC$,I-1,1)=" "ORMID$(SC$,I,1
)=" "THEN2067
2065 IFMID$(SC$,I,1)<"["THENMID$(SC$,I,1
)=CHR$(ASC(MID$(SC$,I,1))+32)
2067 NEXT
2068 I=16:IFSC<1000THENI=17
2070 SC$=LEFT$(SC$,15)
2075 SC$=SC$+STRING$(I-LEN(SC$),32)+STR$
(SC)
2120 PRINT SC$(I)
2141 IFFLTHENPRINT"Press =S= to save new
 scores"
2145 I$=INKEY$
2150 IFI$=CHR$(13)THEN70
2160 IFI$="S"THEN2180
2165 IFI$=CHR$(31)THENFORI=1TO9:SC$(I)="
                        ":SC(I)=0:NEXT:PRIN
T@960,"All but high score CLEARED";:FC=1
2170 GOTO2145
2180 IFFC=0THEN2197ELSEPRINT@704,CHR$(31
)"
So as not to lose the original
HIGH scores, do you want to
change the name?"
2185 I$=INKEY$:IFI$=""THEN2185
2190 IFI$="Y"THENPRINT@704,"
(Orig filespec:  "SC$(11)"
What is the new name";:INPUTSC$(11)
2195 SC$(11)=SC$(11)+STRING$(21-LEN(SC$(
11)),32)
2197 PRINT@768,CHR$(31)"
Adjusting the program..."
2200 FORI=1TO11
2210 SC$(I)=SC$(I)+STR$(SC(I))
2220 FORY=0TO20:POKEP1(I)+Y,ASC(MID$(SC$
(I),Y+1,1))
2230 NEXTY
2240 NEXTI
2245 PRINT"Saving "SC$(11)
2250 SAVESC$(11):RUN
2300 RETURN
```

Rik Pierce
Brooklyn, NY



HAVE WE GOT A PROGRAM FOR YOU IN '82

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

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

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

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

Don't miss the coming of the new computers — show up for the show. Admission is $5 per person and $3 for children.

THE NATIONAL COMPUTER SHOWS

Ticket Information

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

THE SOUTHWEST COMPUTER SHOW

Dallas
Dallas Market Hall

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

DIRECTIONS:
2200 STEMMONS FREEWAY
(AT INDUSTRIAL BLVD)

THE NEW YORK COMPUTER SHOW

Uniondale, Long Island
Nassau Coliseum

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

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

THE TWIN CITIES COMPUTER SHOW

Minneapolis
Minn. Auditorium
& Convention Hall
Third Avenue

Thursday-Sunday
September 16-19, 1982
11 AM to 6 PM Daily

DIRECTIONS: HWY 94 to 11th St. Exit to Third Ave.

THE MID-ATLANTIC COMPUTER SHOW

Washington, DC
DC Armory/Starplex
Across from RFK Stadium

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

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

THE MID-WEST COMPUTER SHOW

Chicago
(Arlington Heights)
Arlington Park Racetrack
Exhibition Center

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

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

THE NORTHEAST COMPUTER SHOW

Boston
Hynes Auditorium/
Prudential Center

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

DIRECTIONS: TAKE MASS PIKE TO PRUDENTIAL CENTER EXIT

THE SOUTHEAST COMPUTER SHOW

Atlanta
Atlanta Civic Center

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

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

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

OUTGOING MAIL

by Nancy Lapointe

It doesn't seem possible that another month has passed. So many things have taken place in a month's time. First, we would like to welcome John G. Grow, the new publisher of ***SoftSide*** Magazine. All of us are excited and anxious to see how John's contributions will enhance ***SoftSide*** and help the magazine expand.

One of the things we have changed, as you will notice on the cover, is that ***SoftSide*** will no longer carry a month date, but rather a number. The number stands for the issue, which will never become outdated. We want people to realize that each issue of ***SoftSide*** is something to keep; a volume, filled with programs, that never becomes outdated. This issue, number 30, represents the first numbered edition of ***SoftSide*** and from now on we will be referring to issue numbers in place of month dates. Now computer and book stores can carry numbered ***SoftSides*** and not be concerned about trying to sell magazines that appear to be out-of-date. ***SoftSide*** never goes out-of-date.

We'd like to give you a behind-the-scenes look and the monthly construction of this magazine we call ***SoftSide.*** It's quite a complicated procedure.

Starting at the beginning, the editorial and programming departments meet to decide on the articles, programs, and reviews we want to include in the issue at hand. Authors have to be assigned articles, products sent to the authors of reviews and, most difficult of all, the programs have to be selected. Picking programs means reviewing a lot of submissions and finding the best ones to bring to you, the reader.

After the material has been selected, the gears start to turn. Time is short and the program bugs start showing up. The authors are, as usual, procrastinating about getting started and the contracts for publishing rights, which have been sent out, haven't come back yet. The panic starts.

Meanwhile, the art department has been given titles and subject ideas so they can start working on the illustrations which accompany the programs; but, for the most part, the art department has to wait for the printed material to arrive from the various departments.

By now the programs for the disk and cassette versions should be on their way to the duplicators; the disks going to one duplicator and the cassettes to another. Before they can actually get started on the duplication, they have to send us verification copies to make sure everything is just right. With any luck, the media copies will be ready and back at ***SoftSide*** in time to be mailed at the same time as the magazine.

Now, if everything is still on schedule, the art department starts putting the puzzle pieces together. Also, let's not forget the copy editor, whose in-basket seems buried. One can sometimes forget that there is a person sitting behind the pasted-up boards stacked on her desk for final edit.

Wait, the ad from this vendor hasn't come in; there is a problem in the translation of the Front Runner and it will take at least three days to work the bug out; the last review should have arrived by Purolator yesterday; the last program contract came back and the author has sold the publishing rights to another publisher. Find another program and start the illustrations immediately. Work the ad in after the proofs come from the printer — The art department is once again burning the midnight oil. With any luck at all, the printer will still get the magazine in enough time for us to get it in the mail on time.

The tension before each issue goes out is often too much to take. The departments are trying to remain on good terms with each other. In three more days the magazine will have gone to print and, until next month, things will seem to quiet down.

The best part in each month's cycle of putting ***SoftSide*** together is the day when that first box of magazines comes back from the printer. All the anxiety seems worthwhile. You see the end result of everyone's efforts in print, along with your name on the mast head, and it almost makes you forget all the frustrations of two weeks ago. You, the reader, are the final judge when you receive your very own copy of ***SoftSide.***

Special issues, like the word processing issue, take almost six months of planning to put together! For all of you waiting for a good time to submit your educational programs we have scheduled Issue #31 — the next issue you'll receive — to be based on educational material and programs. Also, Issue #33, which is ***SoftSide***'s anniversary issue, is going to be based on graphics. We will be publishing high quality graphics programs and reviewing some of the best graphics software on the market. Keep an eye out for these issues for they will be something you won't want to miss. Now is a good time to polish up those educational and graphics programs and send them in for review.

For all of you who either wrote in on the survey or asked at the computer shows about a series of articles on translations, make sure you don't miss this month's initiation of a new *Sensuous Programmer* series. "J" has started, in this issue, describing the procedure for translating from one system to another. Once you become an expert translator, you can take advantage of the offer in this issue to get a one year media subscription for a published translation.

Once again, I've gotten carried away and forgotten that this column has to fit into a certain amount of space. Oh well, I'm sure if it doesn't fit, the art department will haul me in to **make** it fit. It would be nice to hear from some of you regarding your feelings about the last few issues. We always appreciate comments from our readers, positive or negative. It helps us to make ***SoftSide*** a magazine you'll continue to enjoy.

Until next month, Happy Hacking.

Nancy Lapointe

Associate Publisher

ENTERTAINMENT TOMORROW

The Computer Goes to Hollywood

by Fred D'Ignazio and Allen L. Wold

Have you ever seen Walt Disney's *Pinocchio*? Remember when the giant whale Monstro swallowed Pinocchio and held him prisoner in a belly as big as a cathedral?

Or *Fantasia*? Remember the terrible battle between two huge dinosaurs — a Stegosaurus and a Tyrannosaurus Rex? Remember the evil creatures who came alive at night on *Bald Mountain?* Remember poor Mickey Mouse and the dancing broomsticks who tried to drown him in buckets of water?

Did you see *One Hundred and One Dalmatians*? Remember Cruella de Vil? She was the demonic woman who kidnapped the Dalmatian puppies so she could skin them for a fur coat. Remember the chase scene down the mountain, with Cruella's car spitting fire, and Cruella herself looking like an angry fiend from hell?

These were Disney masterpieces. They were full-length animated features — king-sized cartoons. Every character was drawn by hand. Each scene was drawn by hand. Dozens of artists teamed up and worked thousands of hours to produce all the still paintings that, when shown together rapidly, made the cartoon characters come alive.

The pictures were incredibly realistic. The heroes were brave and the villains were frightening and evil. Action filled the screen from start to finish.

But those days are gone. All that's left are the reruns.

Why?

Because full-length, animated cartoons have become tremendously expensive.

The Golden Age of Cartoons

In order to produce a quality animated film, artists need to draw thousands of individual, full-color pictures. Each picture might only be slightly different from the next, but it has to be completely drawn and colored.

Walt Disney Studios pioneered full-color, high-quality animated films. In the 1930s, 1940s, and 1950s, Disney had dozens of animators working turning out such classics as *Sleeping Beauty, Dumbo,* and *Snow White.*

Unfortunately, even though these movies were a great success, costs rose so fast that, after awhile, it became unprofitable to do animated films with any level of detail, craftsmanship, or quality. Almost the only type of animations that remained by the late 1960s and 1970s were the "quick-and-dirty" animations used for Saturday morning cartoon shows.

"In the mid-1980's, thanks to the computer, animation has again become popular."

The End of the Golden Age

The cheaper animations used several shortcuts that significantly decreased their quality. For example, dozens of pictures in a row would be exactly the same, except that the characters' lips would move; or the same chase scene would be repeated several times; or the action of the characters might be jerky and ragged because the animators didn't make enough frames to capture each fine detail of the characters' movements; or the background in the animation might be blank, hastily drawn in, or repetitive.

These shortcuts made for mediocre animations, but they enabled film companies to hire far fewer animators than Disney had working for him during animated films' Golden Age.

Computer Cartoons

In the mid-1980s, thanks to the computer, animation has again become popular. The computer does not produce films automatically. Human artists still do the original artwork, but the computer helps the artist in two major ways. First, the computer helps the artist to work faster. Second, it frees the artist from lots of boring, tedious chores, such as filling in colors and painting the same, exact scene many times in a row.

Now an artist can draw an original picture — with scenery and characters — using a special pen plugged into the computer. As he or she draws, the same picture appears on the computer's TV screen and in the computer's memory.

When a picture is drawn, the artist can choose colors for the characters and scenery by touching a color contained in a "menu" of colors shown on the screen. When the artist touches an area in the picture which is to be painted that color, the computer colors the area automatically.

For example, let's say the artist has just drawn a robot and wants to make it metallic blue. She touches the blue color on the menu and then the robot. Instantly, the computer paints the robot blue.

Or, let's say the robot is crouching with a light saber in its right hand. The scene calls for the robot to leap onto a tall ledge, while holding the saber.

In the past, a good animation film would have required the artist to draw dozens of "inbetweener" pictures that showed the robot in a crouch, flying through the air, and ending up on the ledge. This would have made the robot's jump look natural, lifelike, and realistic.

Now the artist has to draw only two pictures: the robot in the crouch and the robot on the ledge. The computer automatically draws and colors in all the in-between pictures. This saves the artist hours of tedious work, drawing and coloring in the same robot in only slightly different positions.

The World Through the "Eye" of the Computer

Today dozens of companies manufacture computer graphics terminals — computer keyboards with TV screens that make pictures.

Almost all personal computers are built so that you can plug them into your TV and make picutres. Depending on the type of computer or terminal, it might make simple black-and-white stick-figures, or it might paint realistic pictures in full color. Some computer pictures are so realistic that they look like photographs.

Lauren Carpenter is a computer scientist who specializes in teaching computers to draw realistic pictures. He works at Lucasfilm, where he helps to create computer special effects for upcoming episodes in the *Star Wars* saga.

Lauren recently taught his computer to draw a picture of a mountain range that looks exactly like a photo taken by a hardy mountaineer, thousands of feet up the sheer face of a windy, snow-covered mountain. Amazingly, the picture is completely artificial. The "mountains" in the picture are nowhere on earth. They were drawn by a computer program that was trained to manipulate numbers statistically in order to simulate a real mountain. The program creates patterns of light and shadow out of millions of numbers and hundreds of rules, then "paints" what look like mountains on the computer's picture screen.

If Lauren doesn't like the computer's mountains he can erase the screen. A few seconds later, a new mountain range will appear, compliments of the computer's "imagination."

Paint and Grow

Digital paint boxes and computers that can paint imaginary mountain ranges represent the two types of computers used in Hollywood. The first type of computer is a "Paint" machine. It enables a human artist to paint — create and color — film frames electronically. The second type of computer "grows" pictures from scratch, through the use of complex algorithms that create realistic visual images — a combination of subtle textures, contours, tones, shadings, elaborate colors and three dimensional perspectives.

The first type of computer is the tool of a trained human artist or animator. The second type of computer is controlled by a human "artist," too, but not an artist with manual talents and skills. This artist is a technician — a computer scientist or programmer — with knowledge of advanced, statistical programming techniques. This "artist" works only with algorithms, numbers, and equations. The computer actually puts the art onto a screen or a piece of paper. The computer becomes the artist's "hand."

Both types of computers are sometimes employed in the same film.

Animated Electronic Mattes

Advanced computer animation systems are used to create many of the special effects you see in today's science fiction and adventure films. During a live-action filming, movie technicians can project human actors onto scenes created by a computer. The scene is called a matte.

At one time, all mattes were created by human artists. Whenever an especially complex scene was required, a matte was painted, because a real scene would be too time-consuming or expensive to stage.

Even today, many mattes are still stationary paintings. Sections of the mattes are darkened — or masked — so that when the film showing the matte and the film showing the actors are merged, the effect is realistic. Mattes of this type were used in Walt Disney's *Mary Poppins* to show a dazzling rooftop view of the city of London; they were also used to create spectacular scenes in Walt Disney's movie *Black Hole.*

Now, computers can create mattes — mattes that move and have a life of their own. Computers can project living, human actors onto these animated mattes. The scene represented can be anything or anyplace in the universe, or it can come from the film producers' imagination.

Two studios that make extensive use of computer animation systems are Lucasfilm and Disney Productions.

Special Effects for *Star Wars*

Ed Catmull is a wizard at using computers to create special effects for movies. Recently, Ed was hired by Lucasfilm to produce the computerized special effects on future episodes of the *Star Wars* saga.

Ed spent his first year at Lucasfilm getting ready. He built several high-speed animation computers. He designed revolutionary new special effects and he hired the top artists and computer experts in the world.

Now Ed has begun teaching the computers to create animated computer pictures that will come in all colors and in three dimensions. The pictures will be inserted in the midst of live action scenes in upcoming *Star Wars* movies. They will be so exciting and lifelike, that you'll think they are real.

TRON

Walt Disney Productions' movie, *TRON*, is about a video game genius who gets kidnapped to the world inside the computer. MCP, an evil computer program, captures the hero and transports him to a bizarre "game grid" where he becomes an electronic gladiator and must constantly fight battles just to stay alive.

To produce the incredible scenes in *TRON*, Disney hired some of the most famous computer animation companies in the world. These companies taught their computers to create some of the amazing special effects.

Computer Pictures at 4 AM

Computer screens haven't always been able to make pictures. In the 1940s and early 1950s, people used computer screens as part of the computer's memory. Later, in the mid-1950s, people used picture screens as "electronic paper." For example, when a programmer typed in commands or information on the computer typewriter, a copy would appear on the picture screen.

No one knew how to make pictures on a computer screen, only how to enter numbers and words. Everyone knew computers were good at arithmetic, but no one realized that computers could translate numbers into pictures.

Then along came Ivan Sutherland. In the early 1950s Ivan was a graduate student in the Department of Computer Science at MIT. He had been lured to MIT by Artificial Intelligence genius, Marvin Minsky.

Ivan was hooked when Minsky showed him MIT's TX-O computer. The TX-O was an early time-sharing system that allowed a user to interact directly with the computer. Ivan loved sitting down at a terminal and carrying on a conversation with the computer.

When Ivan arrived at MIT, it was summer vacation. Few students and professors were around to compete with him for the computer's time, so he spent all day on the computer. Then the fall semester began, and Ivan got bumped off the computer.

The computer was in use all during the day, but it was free every night after midnight. So Ivan, his wife Marcia, and their two children set up a new schedule for Ivan to follow. At 3:30 every morning, Marcia would get Ivan up, feed him breakfast, and send him off to the computer lab. She'd go back to bed until seven or eight, then get up with the children.

Ivan, meanwhile, would rush over to the lab, sit down at the terminal, and have the computer all to himself until the first professors and computer operators arrived around seven thirty.

The First Electronic Sketchpad

During one early-morning session with the computer, Ivan was playing with the picture screen. He had seen people type dashes on the screen, but no one, he realized, had ever programmed the computer to draw.

Ivan wanted to write a program that would cause the computer to draw a simple line on the picture screen. To do this he had to face several problems that no one else had ever faced.

These are some of the questions Ivan asked himself: How do I describe a line to the computer? What instructions will make the computer actually draw the line? How do I tell the computer where to start the line, and where to stop it? Should the line be of fixed size, like a pole, or should it be elastic, like a rubber band?

For the next two and a half years, Ivan worked on this problem. Finally, he figured out how to make the computer draw a line. Then he found a way to make the computer draw a square, then a triangle. One day, he practically danced around the laboratory. He had made the computer draw a circle!

Ivan didn't stop there. He taught the computer to draw beautiful three dimensional shapes like spheres, cones, and cubes. He taught the computer to spin the shapes round and round. He taught the computer to shrink the shapes and to enlarge them.

Step by step, Ivan blazed a trail into the world of computer graphics. After two and a half years, he proudly unveiled *Sketchpad*, a complete computer picture-making program.

Ivan's enthusiasm about making pictures was contagious. According to Marcia, "Wherever Ivan was, people showed up and got excited. When Ivan was around, things just started happening."

Motion Graphics for the 21st Century

In the next three columns, we will continue to focus on computer graphics, computer animation, and the use of computers to make movies.

We will journey from Ivan Sutherland's early *Sketchpad* system to today's digital paint boxes and animation systems. We will end our journey with an interview of computer genius, John Whitney Jr. We will learn about Whitney's efforts to invent a "Leonardo da Vinci" computer to create totally new modes of electronic entertainment — a "motion graphics for the 21st century."

On the way, we will look at today's *paint* and *grow* computers, and at innovative studios, like Lucasfilm and Disney Productions. We will look at films that have used computer special effects — films like *Star Wars, Alien,* and *TRON.*

We will speculate about interactive "movie theatres" of the future — combinations of computer flight simulators, arcade videogames, and conventional wide-screen movies.

We will look at some of the languages and systems that are being developed, included *Zgrass,* the *Digital Paintbox*, and the powerful animation *Designer's Toolbox.*

We will examine the way computers are enabling scientists and film makers to work together to create a new generation of super-realistic TV documentaries: computer movies with dazzling special effects, all based on scientists' numbers and formulas, and on the latest scientific theories.

With computerized image generation systems we can journey outside the Milky Way, take a spin around the solar system, orbit the earth, or dive through Saturn's rings. We can also soar across history and watch the Big Bang, or witness the fleeting birth and death of a sub-atomic particle. S

The Big Crash

by R. J. Green

Emma Honker was uneasy; her husband, Charlie, had been working for two straight hours at his computer. She couldn't see the screen from where she was sitting, only the pale glow that painted the outline of his shoulders. Things just didn't seem right. Charlie was up to something.

Her eyes fixed on the middle of his back. “What are you doing?” she asked.

The room seemed to fill with colorful light. “Zapp!” shouted Charlie. The screen was covered with an array of flashing colors. In a few seconds it settled down to a colorful throb. “I crashed the system while in BASIC without any pokes,” he said. “I'll bet the boys at Atari® thought it couldn't be done.” Mr. Honker spun around in his seat as if expecting some sort of thunderous applause.

Mrs. Honker returned to her knitting. “I wish you wouldn't do that,” she said. There was silence for five minutes. Again, reflected light from the video screen pulsed around the room.

“Zapp! Zapp!” cried Mr. Honker.

“Please don't do that, Charlie. It makes me nervous. You aren't supposed to blast the system are you?”

“Crash the system,” corrected Charlie.

“Okay, crash the system. You aren't supposed to be doing that are you?”

“That's the whole point, Emma. They designed it to be crash proof but I've done it all the same. I'll bet they'd pay a fortune to know how I did it.”

“You can play with your computer without doing that can't you? Anyone would think you're some sort of kook.” Emma had a particularly irritating way of calling him a kook. She only seemed to call him that when he was having fun.

“You see,” Charlie said evenly, “crashing the system, knowing how to crash the system, is just as important to software design as working the system.”

“Sounds crazy to me,” Emma said, picking up a knitting needle. “I can't see how making it go berserk is any use. It makes me sick just to see it.”

That Thursday the Honkers had dinner at the Milwrights. After dinner they ended up in the family room. Myron poured Charlie a brandy while Theresa and Emma sipped some hot tea. There was an uneasy silence in the room.

Suddenly, Emma had a premonition. She turned around to see what the men were up to. Sure enough, Charlie was seated before Myron's console typing away at the keyboard. Before Emma could say anything, there was a flash on the screen. She could see Charlie clearly, a silly grin covered his face. Then something strange happened. The disk drive started to whir and shudder, then it began to ratchet violently. The drive looked as if it was about to leap from the desk. Myron's face paled as he dove for the switch.

There was silence in the car as Charlie and Emma drove home. Emma waited for him in the kitchen while he stabled the car. “You didn't have to do that,” she said.

“It's all right, Emma,” he said. “They make those disk drives damn sturdy.”

“Myron didn't seem happy with it at all.”

“Myron is too protective.” Charlie began to prepare some instant coffee.

“It was very rude,” Emma said. “I mean, being a guest in someones house then diddling their computer.”

“It's not called diddling,” he said, still in control of himself.

“They probably think you're some sort of nut. A kook,” Emma replied. There was silence until the teakettle came to a boil and began to scream.

From then on things deteriorated rapidly. Late at night Charlie would steal to the family room and work on perfecting his crash program. At such times Emma would lie awake and watch as the glow from the family room became a colorful throb.

“I just can't take it any longer,” she said one day when he came home from work. “If you don't stop crashing the computer, it's all over between us, Charlie.” It was a shock for Charlie to hear it put so bluntly.

“I'd rather not talk about it,” he said as he punched at the TV remote control.

In spite of the temptation, Charlie stopped crashing his computer. Space Invaders soon began to bore him and his backgammon skills progressed well past the computer's capabilities.

One day he began to wonder what the shortest program to crash the system would look like. He started weeding and pruning, trimming the program down. He was merciless. Statements were compressed and code tightened; he was struggling gamely to fit the whole thing into ten lines of code. And then, just when a breakthrough seemed imminent, he felt Emma's presence in the room.

“You can't fool me, Charlie Honker,” she said through clenched teeth.

“Leave me be!” he retorted.

"You don't have to shout," Emma replied. "We won't have a friend left if you don't give this up."

"The hell with friends," Charlie growled. There was a wild look in his eye.

"You're sitting in here like some kook, being destructive. Before long the whole neighborhood will know about it." Emma left the room.

The next evening, after dinner, Emma poured Charlie a drink and sat down next to him. "I was talking to Sally Westfall today."

"Westfall? Doesn't her husband work for Atari®?" Charlie asked.

"Yes, anyway, she was telling me about this doctor. He's done wonders for Bobby. Why in no time at all, the boy was playing baseball again."

"What in God's name are you up to now, Emma?"

"Nothing. I just thought...well, he's sort of an expert in these things — you know, obsessive behavior and all."

Charlie went to the family room and seated himself. The faithful 810 ground away as Charlie mumbled "Mad am I? I'll show them who's mad. There are depths to this thing they haven't even dreamed of."

Emma followed him to the family room. "Dr. Laskey says that home computers are like a disease. He says this sort of thing is becoming epidemic."

"Epidemic, my foot," Charlie said, turning up the audio. "I'll bet he doesn't know a peek from a poke."

"You're up to it again aren't you? You're going to crash the system again. Admit it."

"Crashing the system is child's play. If you really want to know, I'm figuring out how to double flip the video display."

It must be said on Charlie's behalf that his words were delivered very calmly, very evenly, and very slowly. There was only a hint of emotion in the statement. They both had a vague feeling of being trapped on a runaway roller coaster.

That night, while Emma lay in bed studying the ceiling, Charlie was at the keyboard. "Flip, Flip," he said. The characters on the screen not only reversed themselves but also formed themselves upside down.

The next morning Emma found Charlie slumped across his desk. The screen was pulsing hideously above him in a frenzy of dancing letters and symbols.

"My God, it's killed him," she screamed.

Some ten minutes later, Emma had recovered enough to call Dr. Laskey. Laskey responded quickly and was at the Honker's door within the hour. He was getting used to these calls. They were coming more and more frequently the last few months. Would people never learn?

Charlie was just regaining consciousness as the doctor rushed into the room. Laskey recognized the telltale signs revealed between Charlie's half closed eyelids: the shrunken pupils, and the glazed irises. Charlie's fingers were curling and uncurling in a ghastly parody of a pianist flexing his fingers before a concert. "This is the worst case I've ever seen," Laskey said, carefully polishing the right lens of his glasses. "The man is as mad as a hatter."

It took ten long months of grim therapy before Charlie was allowed to leave the rest home. In the interim, Emma joined a religious community somewhere in Kansas. She now dresses in black and refuses to ride in automobiles.

Charlie is seldom seen anymore. His job is gone and he spends his days locked in a cheap apartment. Occasionally his neighbors call the police with complaints of strange whirring noises and flashes of light radiating out from underneath his door.

SOFTSIDE ORDERING INFORMATION

FORM OF PAYMENT

USA

VISA, MasterCard, certified checks, money orders and personal checks are accepted.

Canada/Mexico

The preferred method of payment is by VISA or MasterCard. A bank check is acceptable if it has been preprinted for payment in U.S. dollars. No personal or company checks accepted.

Other Foreign Orders

Payment must either be by a bank check drawn on a U.S. bank payable in U.S. dollars or by affiliated bank credit cards of VISA or MasterCard.

GUARANTEE

All software is guaranteed to load and run. If you experience difficulties with the product within 30 days, it may be returned for replacement. Send your properly protected tape or disk to the attention of the Customer Service Representative and include your name, address, and the reason it is being returned.

LIABILITY

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

PRICES

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

Unless otherwise noted in a published advertisement, the following prices are in effect as of this issue:

	USA	Mexico APO/FPO Canada	Other Foreign
SoftSide Magazine (yr)	$30	$40	$62
CV (year)	$75	$95	$125
(6 mo.)	$39	n/a	n/a
DV (year)	$125	$145	$175
(6 mo.)	$64	n/a	n/a
Adventure of the Month Month (6 mo.)			
Cassette	$29	$35	$41
Disk	$49	$55	$61
Parsec	$20	$21.50	$25
Envyrn	$200	$202	$210
Diversions	$60	$70	$85

BACK ISSUES

Minimum order for magazines only — 3 issues. There is no minimum order for magazine/media combinations.

Price includes shipping to the 48 states only. Alaska, Hawaii, Puerto Rico, APO/FPO, and ALL foreign orders — postage is additional.

ALL Foreign orders and all magazine/media combination orders — Order directly from **SoftSide, 6 South St., Milford, NH 03055.**

CALENDAR

June 4-6
The First Annual New Hampshire Computer and Business Show
National Guard Armory, Manchester, NH

June 4 and 5 10 A.M.-7 P.M.
June 6 10 A.M.-5 P.M.
This show will bring makers and distributors of personal computers, their accessories, software and related business equipment to one place — the National Guard Armory — to show and demonstrate their equipment to thousands of pre-qualified buyers who will pay to attend.

June 5
Amateur Fair
Minnesota State Fairgrounds, St. Paul, MN

The North Area Repeater Association will sponsor Minnesota's largest swapfest and exposition of personal computer and communications equipment. Show will feature exhibits, booths, and prizes. Admission is $3.00.
Contact: Amateur Fair, P.O. Box 30054, St. Paul, MN 55175.

June 6-8
Sixth International Conference on Computers and the Humanities
McKimmon Center, Raleigh, NC

A pre-convention workshop will be presented. Speakers will be featured during the conference.
Contact: Department of English, North Carolina State University, Raleigh, NC 27650.

June 7-10
National Computer Conference
Astrodomain, Houston, TX

The latest advances in computer hardware, software and services will be displayed. This event will also include a technical session, speakers from both industry and government, and films relating to computer technology.
Contact: American Federation of Information Processing Societies, 1815 N. Lynn St., Arlington, VA 22209, (703) 558-3610.

June 8-9
Confronting the Communications Revolution
McGraw-Hill World Headquarters Conference and Exposition Center, New York, NY

Current and future industry changes will be discussed. Hands-on strategy workshops will concern broadcasting, newspaper, consumer and trade publications, and information systems.
Contact: Business Week Executive Programs, 1221 Ave. of the Americas, 40th Floor, New York, NY 10020, (212) 997-4930.

June 14-16
The First Annual Connecticut Business and Personal Computer Show
Hartford Civic Center, Hartford, CT

June 14th and 15th 10 A.M.-8 P.M.
June 16 10 A.M.-5 P.M.
This show will bring makers and distributors of personal computers and their peripherals, accessories, and software to one place — the Civic Center — to show and demonstrate their equipment to thousands of pre-qualified buyers who will pay to attend.

June 20-Aug 6
Young People's Basic Training Camps
Lake Forest College, Lake Forest, IL

These camps will give high school aged students hands-on experience with computers. Camps are one week long and are open to students 12 to 18 years old.
Contact: Lake Forest Computer Camp, Lake Forest College, Lake Forest, IL 60045.

June 27
Rocky Mountain Computer Camps
Boulder Computer Resource Center, Boulder, CO

This summer the Boulder Computer Resource Center is offering computer camps for kids: The ROCKY MOUNTAIN COMPUTER CAMPS. The camps, to be held at Wild Basin Lodge in the mountains 65 miles west of Denver, are dedicated to teaching today's young people skills and tools that will assist them in meeting the challenges of tomorrow's world.
Rocky Mountain Computer Camps will be offered three times during the summer. The first session starts on Sunday, June 27, 1982. For further information and a camp brochure contact BCRC at 1005 Pearl Street, Boulder, CO 80302, 442-6228.

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

SoftSide Publications
Calendar Editor
6 South Street
Milford, NH 03055

Be sure to include complete information concerning dates, location, subject matter and a contact name, address, and phone number.

Bugs, Worms, and other Undesirables

A small bug in the Apple version of *Database* (October, 1981) keeps the custom print formatting feature from working properly. Line 3830 should be corrected as follows:

```
3830  IF J1 < 5 THEN N =  VAL ( MID$
      (F$(T),J,2)):J = J + 2
```

The following minor corrections should be made to the Apple Version of *Word Wars* (January, 1982):

```
1335  GOSUB 255: GOTO 1355
1350  GOSUB 255: GOTO 1365
```

In the ATARI® program *Defense* (February, 1982), a character was dropped from the listing in line 10040. There should be a left parenthesis following the word PEEK.

An important portion of the documentation for *Gothic Letter Printer* (April, 1982) was omitted from the magazine. The character assigned to C$ in line 550 may need to be changed to work properly with your printer. On the Epson MX-80, for example, the CHR$(127), which is specified, will not print anything. You can experiment with the value of this character to change the appearance of the final printout.

An additional note: DV subscribers and others who are working under DOS PLUS will need to change the FORMS parameters to prevent the printout from breaking to skip over page boundaries. From DOS, type

FORMS (L=66)

and press ENTER; this will allow all 66 lines per page to be printed.

There is a small error in the Apple K-Byter *Battleship*, in the April, 1982 issue. Line 160 should read as follows:

```
160  IF AX < 0 OR AX > 9 OR DY <
     0 OR DY > 9 THEN 150
```

SENSUOUS PROGRAMMER

Anatomy 101

by "J"

When I closed my twelfth (and supposedly last) column two issues ago, I had no idea how prophetic the concluding sentence would be. Sigh.

For those who can't remember (or never read) those immortal words, they were a paraphrase of author Alfred P. Morgan's comment at the close of his classic book which so influenced my formative years: "And who knows but that someday, perhaps I shall write another *Boy Electrician*." Mr. Morgan took the sensible course of leaving that sparkling hope dangling enticingly in the air. Fool that I am, I have once again put fingertips to keyboard to continue the legacy of the Sensuous Programmer, with further meanderings through the world of BASIC programming.

Judging from the mail that ***SoftSide*** receives, there are a great many people who read this magazine because they like the multi-system emphasis and are interested in learning more about all three systems. An increasing number of you own or use two or more different computers at home or at work, and want practical help with converting BASIC programs from one to another. (My personal suspicion is that most of these people are TRS-80® owners who sneak in to play with the Apple at work during their coffee breaks.)

In previous columns, I've touched on most of the different kinds of information needed to do such conversions. What is now needed is a step-by-step, how-to-do-it approach, in order to clarify and apply this information. It's one thing to read an article comparing the ways in which the Apple, ATARI®, and TRS-80®handle strings, screen formatting, or graphics; it's something else to actually take a program written in one dialect of BASIC and translate it for another machine.

This is especially true if you have only a program listing of the original, and no access to the computer for which it was written. Trying to visualize how a program will act and what the screen displays will look like, just from a listing, can be a very difficult and frustrating task. Thorough documentation is a big help here, as is the logical structuring of the original program into meaningful sections and subroutines. But those are things you can't always expect to find.

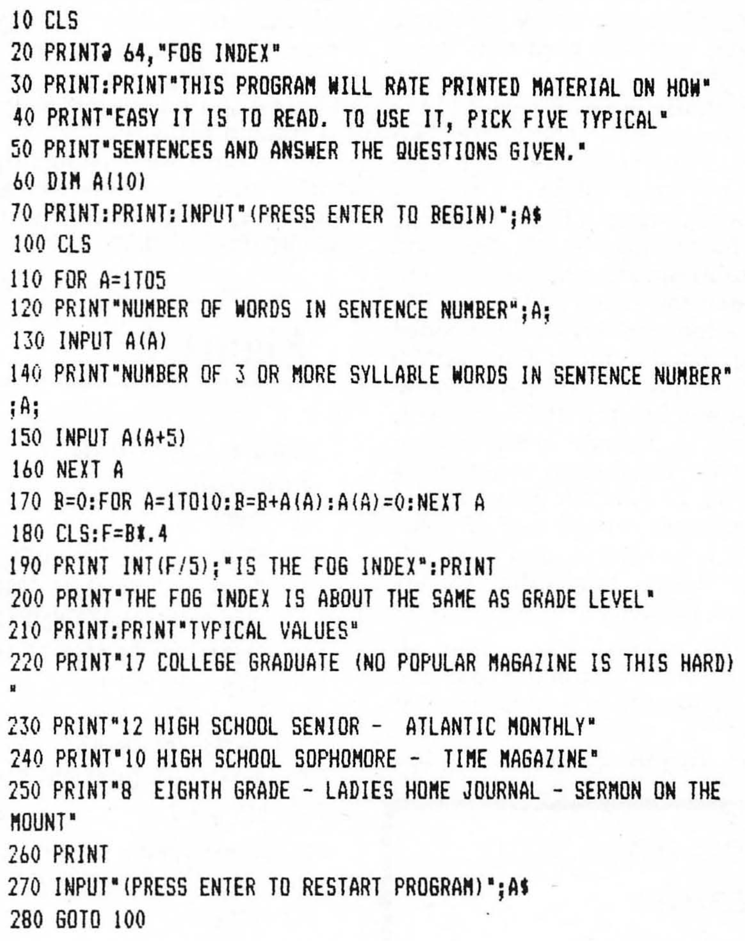
I think the only approach to such a practical subject is practice. In order to learn how to do something, you have to do it. (This bit of wisdom was ingrained in my gray matter as far back as elementary school, when we all sat around thinking up book titles such as *Brain Surgery Self-Taught* by Les I. Q. Moorhead.) So, beginning with the simple and working toward the complex, this series of columns will look at actual programs written for one computer, and will intimately explore the anatomy of translating them for the other two systems. This, of course, is practice for ME, not for YOU. The ultimate step, of applying such fuddlings and fumblings to that program that you want to convert, is up to you.

In this introductory column, I'd like to encourage you to send in your suggestions for programs (or interesting subroutines or portions of programs) to translate here in this series. Feel free to suggest programs of particular interest that have been published in past issues of ***SoftSide***, or others that your computer may have lying around its room...er, RAM. If you do send a program, be sure to send it on disk or tape, not just as a printed listing. Due to the volume of mail I anticipate getting, I'm sure I won't be able to respond personally to each one of you (that's the standard excuse, isn't it?), but I will try to include material of general interest in the column. Tapes and disks will be returned if accompanied by a self-addressed, stamped envelope and a reasonable bribe. Send all materials to me c/o ***SoftSide***, 6 South Street, Milford, NH 03055.

For openers, we'll resuscitate a TRS-80® program from the early days of ***SoftSide*** and step through its conversion for Apple and ATARI®. The program is *Fog Index* by George Blank (March, 1979), and its listing is shown in figure 1. It is very straightforward BASIC code, which will provide an illustration of how easy conversion can be.

Figure 1

TRS-80® Original of *Fog Index*



```
10 CLS
20 PRINT@ 64,"FOG INDEX"
30 PRINT:PRINT"THIS PROGRAM WILL RATE PRINTED MATERIAL ON HOW"
40 PRINT"EASY IT IS TO READ. TO USE IT, PICK FIVE TYPICAL"
50 PRINT"SENTENCES AND ANSWER THE QUESTIONS GIVEN."
60 DIM A(10)
70 PRINT:PRINT:INPUT"(PRESS ENTER TO BEGIN)";A$
100 CLS
110 FOR A=1TO5
120 PRINT"NUMBER OF WORDS IN SENTENCE NUMBER";A;
130 INPUT A(A)
140 PRINT"NUMBER OF 3 OR MORE SYLLABLE WORDS IN SENTENCE NUMBER"
;A;
150 INPUT A(A+5)
160 NEXT A
170 B=0:FOR A=1TO10:B=B+A(A):A(A)=0:NEXT A
180 CLS:F=B*.4
190 PRINT INT(F/5);"IS THE FOG INDEX":PRINT
200 PRINT"THE FOG INDEX IS ABOUT THE SAME AS GRADE LEVEL"
210 PRINT:PRINT"TYPICAL VALUES"
220 PRINT"17 COLLEGE GRADUATE (NO POPULAR MAGAZINE IS THIS HARD)
"

230 PRINT"12 HIGH SCHOOL SENIOR -  ATLANTIC MONTHLY"
240 PRINT"10 HIGH SCHOOL SOPHOMORE -  TIME MAGAZINE"
250 PRINT"8  EIGHTH GRADE - LADIES HOME JOURNAL - SERMON ON THE
MOUNT"
260 PRINT
270 INPUT"(PRESS ENTER TO RESTART PROGRAM)";A$
280 GOTO 100
```

The first step in doing a translation is to understand WHAT the program does. (Forget HOW it does it, for the moment.) Presumably even the most sketchy documentation ought to give some kind of description; otherwise, why would you want to go to the work of translating the program in the first place? *Fog Index* helps you to do a brief analysis of a sample of written text, to determine how easy it is to read. You choose five sentences from the text you want to analyze, and the program prompts you to enter two facts about those sentences: the total number of words in each, and the number of three-or-more-syllable words in each. It then gives you a rough idea of the grade level of the writing, and prompts you to press a key to run another analysis.

The next step in the translation process is to scan the program listing to get an overall idea of its complexity and its structure. In this case, the programming is very straightforward, using only the standard keywords PRINT, INPUT, DIM, FOR/TO/NEXT, and GOTO, plus the TRS-80® keyword CLS. Instructions and array dimensioning are right at the beginning (lines 10-70), followed by the main body of the program. There are only two FOR/NEXT loops: One for inputting information about the five sentences in lines 110-160, and one in line 170 for doing the calculation of the fog index. Other than these, and the GOTO 100 at the end which simply returns to the beginning of the main body, the program just flows from line to line without branching off to foreign line numbers.

With this overall structure in mind, it's a good idea to look more closely at the coding, with an eye to spotting unfamiliar keywords, or familiar ones used in unfamiliar ways. The very first line of the program contains a statement which may be unfamiliar to an Apple or ATARI® user: CLS, which simply clears the screen. It's the equivalent of HOME or CALL -936 on the Apple, or PRINT CHR$(125) or PRINT"(esc ctrl-clear)" on the ATARI®.

Line 20 may also contain an unfamiliar element: the "@" symbol following the word PRINT. This combination directs printing to the specified location on the screen. The location can be between 0 and 1023; the first screen line is numbered 0-63 from left to right, the second 64-127, and so forth. Printing at location 64, then, begins printing with the first character on the second line of the screen. This function can be duplicated by ATARI®'s POSITION statement or Apple's VTAB, or (in this case) simply by printing a blank line before the actual text.

Lines 30-50 are just text to print; what could be more straightforward? Yet they will require some modification because they're formatted for 64-character lines, not 38 or 40. Lines that break in the middle of words are SO tacky; and yet translators sometimes gloss over such "trivial" details, ruining the final appearance of the converted program. The easy way to properly format text for your computer is simply to pay attention as you're typing in the PRINT statements. When the string of text wraps around so that it is directly under the opening quotation mark, you're at the last character of the screen line; the next line will begin immediately under the first character of the line above. (Notice, however, that the Apple does not LIST program lines in this way once they are entered into memory; you need to pay attention as you're typing the line.) You can choose either to use a separate PRINT statement for each screen line, or to combine two or more screen lines in a single PRINT, keeping the first character of each new line directly under the previous one.

The DIM statement in line 60 needs no conversion, since all three computers handle numeric arrays in the same way. But, when comparing the

Apple or TRS-80® with the ATARI®, statements which dimension strings are another matter altogether. This subject will be covered in a later column, in the context of another program.

The remainder of the program will cause no problems at all for the Apple translation, and will need only some slight modifications for the ATARI® translation with regard to the INPUT statements. There are two types of structures here that the ATARI® will not allow: Using a prompt string between INPUT and the variable name (lines 70 and 270), and using a subscripted variable in the INPUT statement (lines 130 and 150). Both situations are very easily re-coded into an acceptable form. For lines 70 and 270, simply use a PRINT statement followed by an INPUT statment, being sure to include a DIMension statement for the dummy input string before using it. And for lines 130 and 150, use a simple variable such as A for the INPUT statement, and then immediately assign this value to the subscripted variable.

With the original listing now all scribbled up with notes and arrows and such, all that needs to be done is to sit down and start typing. The results are shown in figures 2 and 3. A few modifications having to do with screen formatting (extra PRINTs and that sort of thing) were made in both versions. This sort of fine-tuning is invariably necessary, and can only be done after you have the program running and can see how it looks on the screen. You can now type the appropriate coding into your computer, and use it to find out just how obscure this column really is.

Next month we'll plunge right into another anatomy lesson, with a slightly more complicated specimen to further test our surgical skills.

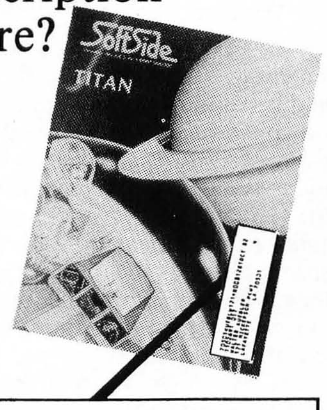


Figure 2

Apple Version of *Fog Index*

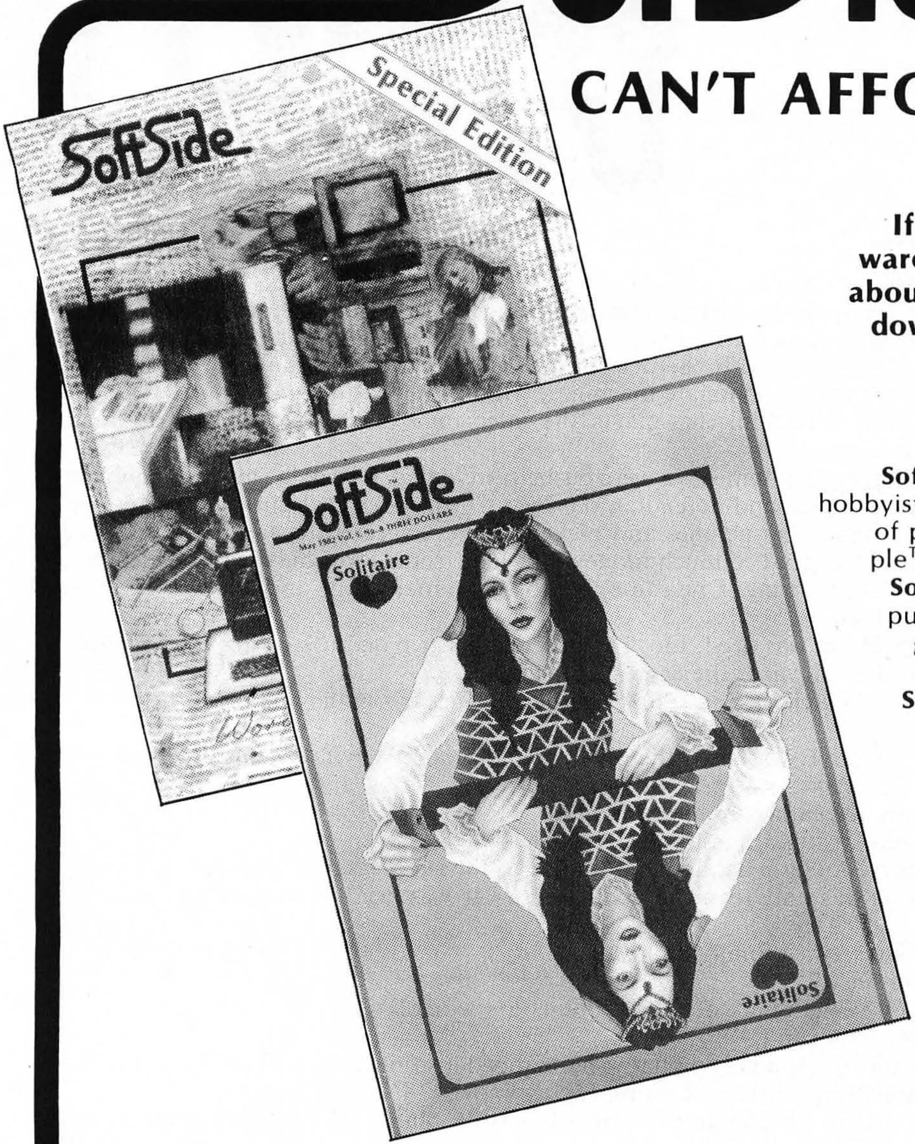
```
10  HOME
20  PRINT : PRINT "FOG INDEX"
30  PRINT : PRINT "THIS PROGRAM W
     ILL RATE PRINTED MATERIAL"
40  PRINT "ON HOW EASY IT IS TO R
     EAD.  TO USE IT,"
50  PRINT "PICK FIVE TYPICAL SENT
     ENCES AND ANSWER"
55  PRINT "THE QUESTIONS GIVEN."
60  DIM A(10)
70  PRINT : PRINT : INPUT "(PRESS
     RETURN TO BEGIN)";A$
100  HOME
110  FOR A = 1 TO 5
120  PRINT "# OF WORDS IN SENTENC
     E ";A;
130  INPUT A(A)
140  PRINT "# OF 3+ SYLLABLE WORD
     S IN SENTENCE ";A;
150  INPUT A(A + 5): PRINT
160  NEXT A
170 B = 0: FOR A = 1 TO 10:B = B +
     A(A):A(A) = 0: NEXT A
180  HOME :F = B * .4
190  PRINT  INT (F / 5);" IS THE
     FOG INDEX": PRINT
200  PRINT "THE FOG INDEX IS ABOU
     T THE SAME AS GRADELEVEL"
210  PRINT : PRINT "TYPICAL VALUE
     S"
220  PRINT : PRINT "17  COLLEGE G
     RADUATE (NO POPULAR
        MAGAZINE IS THIS HARD)"
230  PRINT : PRINT "12  HIGH SCHO
     OL SENIOR (ATLANTIC
        MONTHLY)"
240  PRINT : PRINT "10  HIGH SCHO
     OL SOPHOMORE (TIME
        MAGAZINE)"
250  PRINT : PRINT " 8  EIGHTH GR
     ADE (LADIES HOME JOURNAL,
        SERMON ON THE MOUNT)"
260  PRINT
270  INPUT "(PRESS RETURN TO REST
     ART PROGRAM)";A$
280  GOTO 100
```

Figure 3

ATARI® Version of *Fog Index*

```
10 PRINT "}"
20 PRINT :PRINT "FOG INDEX"
30 PRINT :PRINT "This program will rat
e printed"
40 PRINT "material on how easy it is t
o read."
50 PRINT "To use it, pick five typical
 sentencesand answer the questions giv
en."
60 DIM A(10),A$(10)
70 PRINT :PRINT :PRINT "(PRESS RETURN
TO BEGIN)";:INPUT A$
100 PRINT "}"
110 FOR A=1 TO 5
120 PRINT "# of words in sentence ";A;
130 INPUT AA:A(A)=AA
140 PRINT "# of 3+ syllable words ";
150 INPUT AA:A(A+5)=AA
160 NEXT A
170 B=0:FOR A=1 TO 10:B=B+A(A):A(A)=0:
NEXT A
180 PRINT "}":F=B*0.4
190 PRINT INT(F/5);" IS THE FOG INDEX"
:PRINT
200 PRINT "The Fog Index is about the
same as    grade level"
210 PRINT :PRINT "TYPICAL VALUES"
220 PRINT :PRINT "17  College graduate
 (no popular          magazine is this
 hard)"
230 PRINT :PRINT "12  High school seni
or (Atlantic          Monthly)"
240 PRINT :PRINT "10  High school soph
omore (Time           Magazine"
250 PRINT :PRINT " 8  Eighth grade (La
dies Home Journal,    Sermon on the Mo
unt)"
260 PRINT
270 PRINT "(PRESS RETURN TO RESTART PR
OGRAM)";:INPUT A$
280 GOTO 100
```

SoftSide™



CAN'T AFFORD SOFTWARE? READ THIS:

If the high price of commercial software and the lack of clear information about your microcomputer has got you down, here's the solution you've been waiting for!

SoftSide Magazine

SoftSide is a favorite of computer users and hobbyists alike. They rely on it as a prime source of programs, reviews and articles for the Apple™, ATARI®, and TRS-80® microcomputers. **SoftSide** is the magazine for the microcomputer owner who wants to learn BASIC programming, learn MORE about BASIC programming, or just wants to have FUN! **SoftSide** gives you the BASIC code listings of several programs — adventures, utilities, games, simulations, you name it — for your computer EVERY MONTH.

There's more:

- **Reviews** — of the software and hardware products you want to know about.
- **Articles** — about all aspects of BASIC programming.
- **Tutorials** — on graphics, use of important commands, and more.
- **Programs** — each month **SoftSide** publishes a variety of program for the Apple, ATARI® and TRS-80®.
- **Columns** — which discuss different topics including: computer graphics, picking the right modem for you and marketing your software — just to name a few.
- **Input from our readers** — each month we devote a space in the magazine to let our readers give us some feedback about **SoftSide**.
- **Hints & Enhancements** — programmers and readers provide us with enhancements, to our programs, and programming tips.

Use coupon to order. Mail to: **SoftSide** Publications, 6 South St., Milford, NH 03055

As you can see, you'll receive pages and pages of information and entertainment from SoftSide. Here's the best part:

A subscription to SoftSide is just $24 a year. That's 12 issues for only $2 each! What a value!

□ **YES! Send me the first copy of my SoftSide subscription right away!**

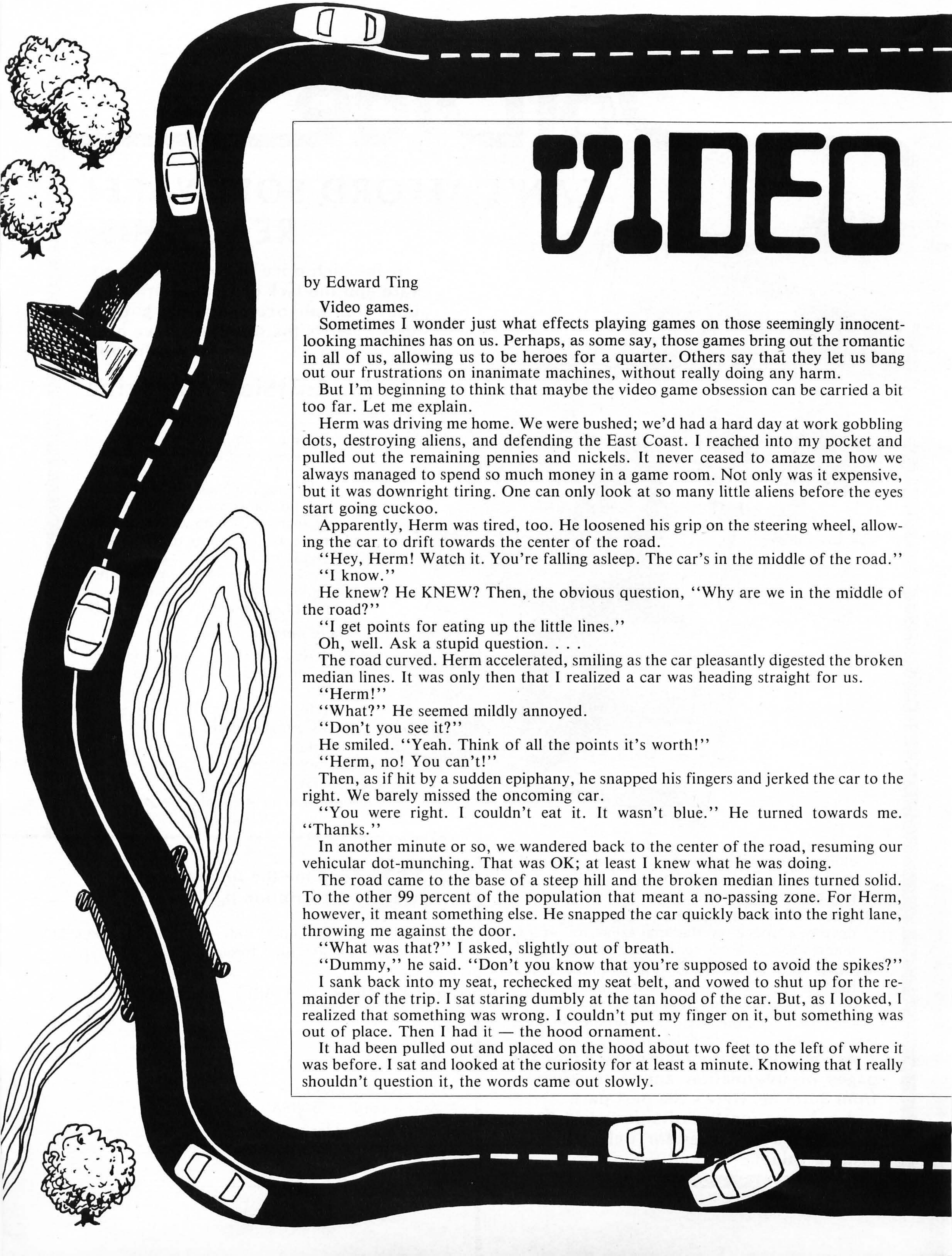
$24/yr for USA and Canada only. For orders to APO/FPO or Mexico — $40/yr. Other foreign orders — $62/yr.

I own a □ Apple □ ATARI® □ TRS-80®

Name ______________________
Address ______________________
City/State ______________ Zip________

□ Check is enclosed
□ MasterCard □ VISA

Name of Cardholder ______________________
MC# and Interbank#/VISA# ______________________
Exp. Date ______________________
Signature ______________________



VIDEO

by Edward Ting

Video games.

Sometimes I wonder just what effects playing games on those seemingly innocent-looking machines has on us. Perhaps, as some say, those games bring out the romantic in all of us, allowing us to be heroes for a quarter. Others say that they let us bang out our frustrations on inanimate machines, without really doing any harm.

But I'm beginning to think that maybe the video game obsession can be carried a bit too far. Let me explain.

Herm was driving me home. We were bushed; we'd had a hard day at work gobbling dots, destroying aliens, and defending the East Coast. I reached into my pocket and pulled out the remaining pennies and nickels. It never ceased to amaze me how we always managed to spend so much money in a game room. Not only was it expensive, but it was downright tiring. One can only look at so many little aliens before the eyes start going cuckoo.

Apparently, Herm was tired, too. He loosened his grip on the steering wheel, allowing the car to drift towards the center of the road.

"Hey, Herm! Watch it. You're falling asleep. The car's in the middle of the road."

"I know."

He knew? He KNEW? Then, the obvious question, "Why are we in the middle of the road?"

"I get points for eating up the little lines."

Oh, well. Ask a stupid question. . . .

The road curved. Herm accelerated, smiling as the car pleasantly digested the broken median lines. It was only then that I realized a car was heading straight for us.

"Herm!"

"What?" He seemed mildly annoyed.

"Don't you see it?"

He smiled. "Yeah. Think of all the points it's worth!"

"Herm, no! You can't!"

Then, as if hit by a sudden epiphany, he snapped his fingers and jerked the car to the right. We barely missed the oncoming car.

"You were right. I couldn't eat it. It wasn't blue." He turned towards me. "Thanks."

In another minute or so, we wandered back to the center of the road, resuming our vehicular dot-munching. That was OK; at least I knew what he was doing.

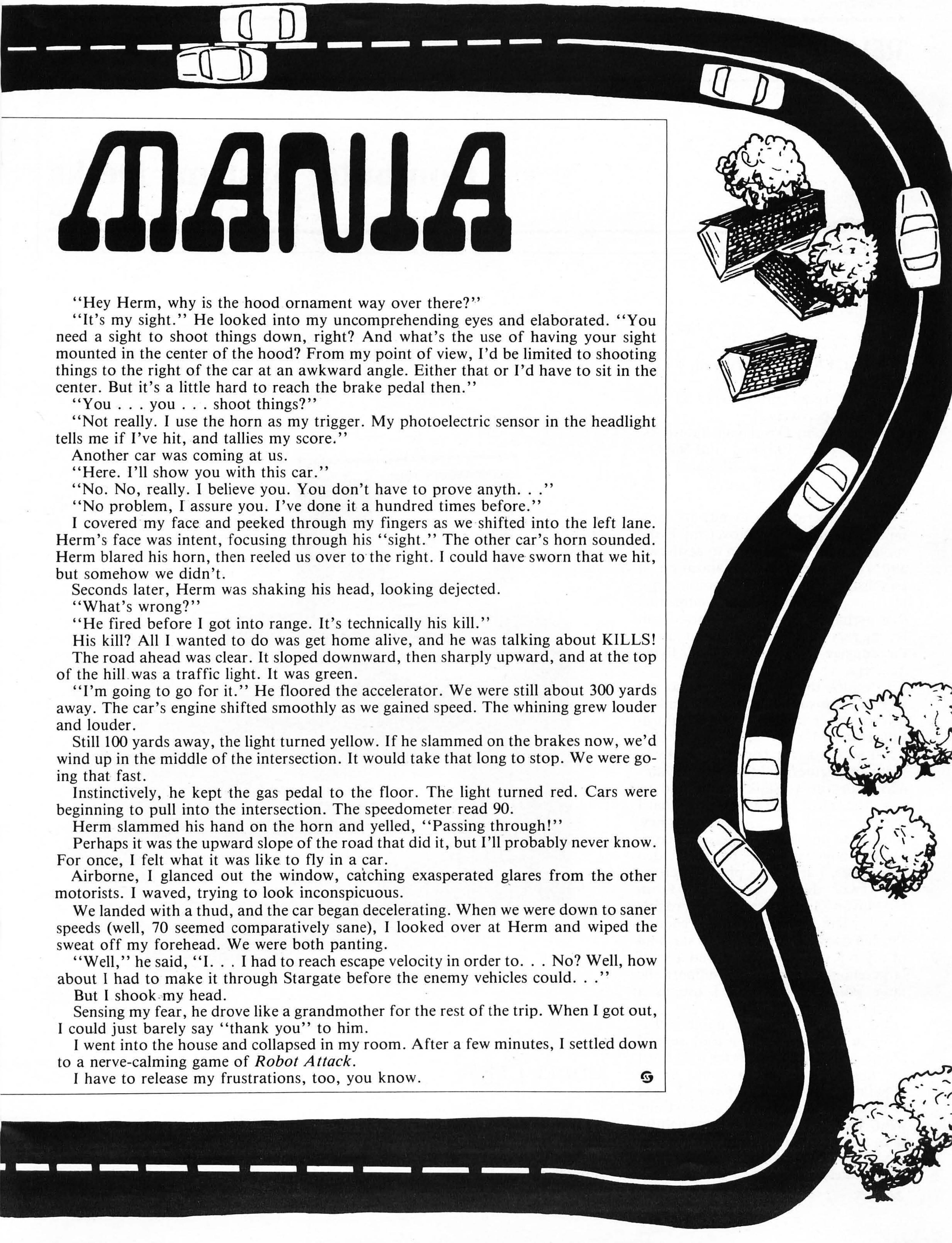
The road came to the base of a steep hill and the broken median lines turned solid. To the other 99 percent of the population that meant a no-passing zone. For Herm, however, it meant something else. He snapped the car quickly back into the right lane, throwing me against the door.

"What was that?" I asked, slightly out of breath.

"Dummy," he said. "Don't you know that you're supposed to avoid the spikes?"

I sank back into my seat, rechecked my seat belt, and vowed to shut up for the remainder of the trip. I sat staring dumbly at the tan hood of the car. But, as I looked, I realized that something was wrong. I couldn't put my finger on it, but something was out of place. Then I had it — the hood ornament.

It had been pulled out and placed on the hood about two feet to the left of where it was before. I sat and looked at the curiosity for at least a minute. Knowing that I really shouldn't question it, the words came out slowly.



MANIA

"Hey Herm, why is the hood ornament way over there?"

"It's my sight." He looked into my uncomprehending eyes and elaborated. "You need a sight to shoot things down, right? And what's the use of having your sight mounted in the center of the hood? From my point of view, I'd be limited to shooting things to the right of the car at an awkward angle. Either that or I'd have to sit in the center. But it's a little hard to reach the brake pedal then."

"You . . . you . . . shoot things?"

"Not really. I use the horn as my trigger. My photoelectric sensor in the headlight tells me if I've hit, and tallies my score."

Another car was coming at us.

"Here. I'll show you with this car."

"No. No, really. I believe you. You don't have to prove anyth. . ."

"No problem, I assure you. I've done it a hundred times before."

I covered my face and peeked through my fingers as we shifted into the left lane. Herm's face was intent, focusing through his "sight." The other car's horn sounded. Herm blared his horn, then reeled us over to the right. I could have sworn that we hit, but somehow we didn't.

Seconds later, Herm was shaking his head, looking dejected.

"What's wrong?"

"He fired before I got into range. It's technically his kill."

His kill? All I wanted to do was get home alive, and he was talking about KILLS!

The road ahead was clear. It sloped downward, then sharply upward, and at the top of the hill was a traffic light. It was green.

"I'm going to go for it." He floored the accelerator. We were still about 300 yards away. The car's engine shifted smoothly as we gained speed. The whining grew louder and louder.

Still 100 yards away, the light turned yellow. If he slammed on the brakes now, we'd wind up in the middle of the intersection. It would take that long to stop. We were going that fast.

Instinctively, he kept the gas pedal to the floor. The light turned red. Cars were beginning to pull into the intersection. The speedometer read 90.

Herm slammed his hand on the horn and yelled, "Passing through!"

Perhaps it was the upward slope of the road that did it, but I'll probably never know. For once, I felt what it was like to fly in a car.

Airborne, I glanced out the window, catching exasperated glares from the other motorists. I waved, trying to look inconspicuous.

We landed with a thud, and the car began decelerating. When we were down to saner speeds (well, 70 seemed comparatively sane), I looked over at Herm and wiped the sweat off my forehead. We were both panting.

"Well," he said, "I. . . I had to reach escape velocity in order to. . . No? Well, how about I had to make it through Stargate before the enemy vehicles could. . ."

But I shook my head.

Sensing my fear, he drove like a grandmother for the rest of the trip. When I got out, I could just barely say "thank you" to him.

I went into the house and collapsed in my room. After a few minutes, I settled down to a nerve-calming game of *Robot Attack*.

I have to release my frustrations, too, you know.

Computer Systems Desks

Reviewed by Randal L. Kottwitz

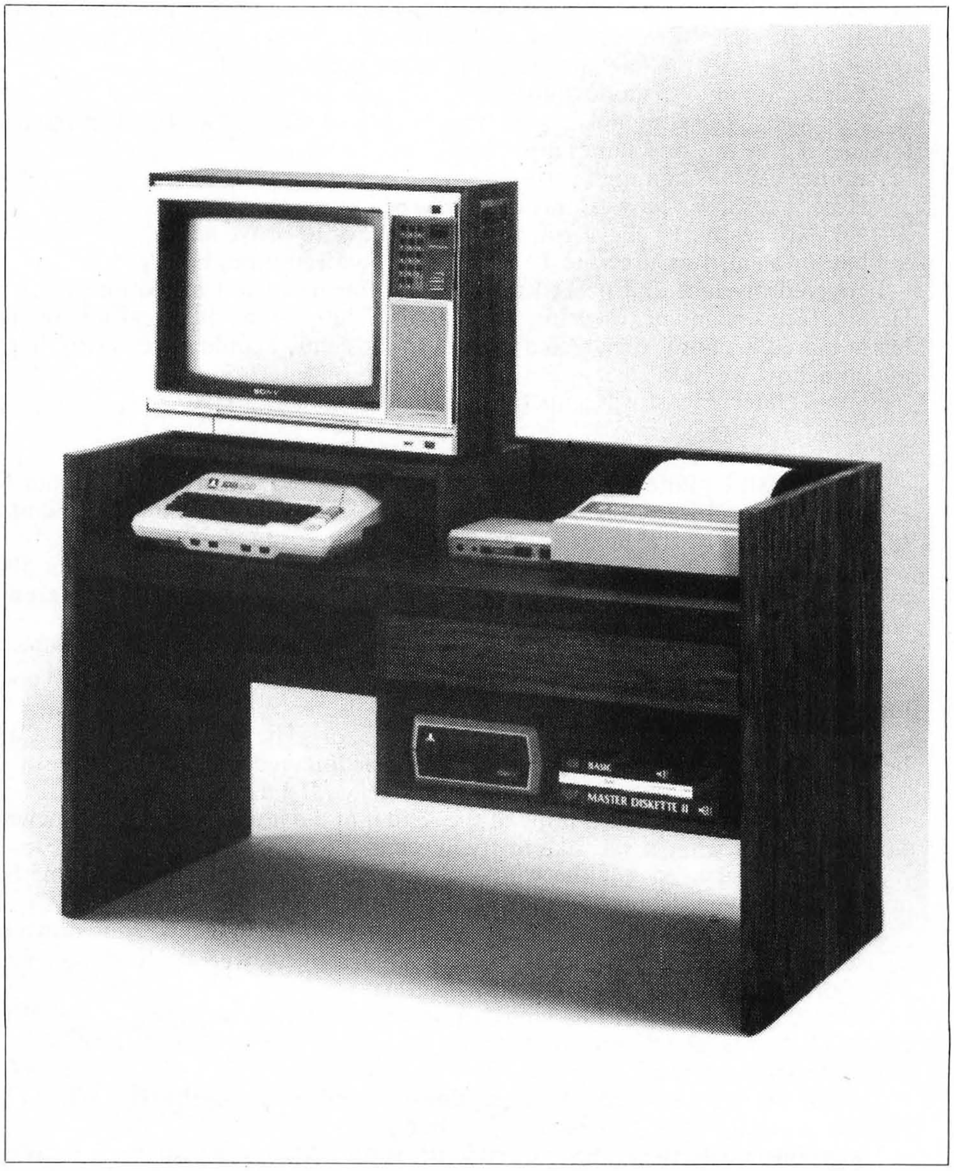
Models CT707 (smaller) and CT709 (larger)
Suggested retail price: $119.95 and $149.95 respectively.
Available from: O'Sullivan Industries
19th and Gulf Streets
Lamar, Missouri
64759

I was desperate! Bit by bit, my computer system had taken over my living room. It paid no attention to aesthetics and the pattern of organization it established for itself was nothing but inefficient. My neighbors commented that perhaps it was time I started collecting rent from the machine to justify the damage it had done to my living quarters.

I decided the time had come to put an end to this injustice and to confine my computer system to the organization I wanted.

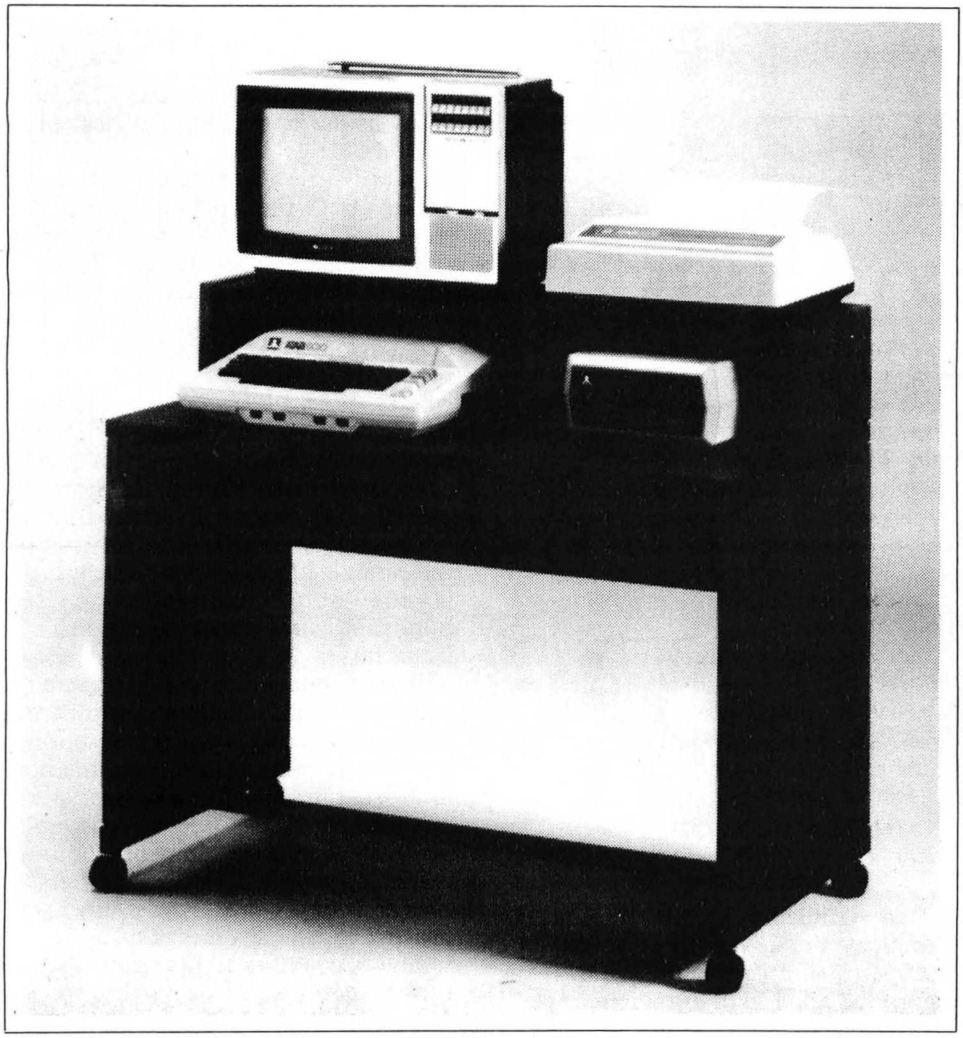
I'd seen ads for computer desks in many magazines, but they either didn't allow for the versatility I desired or they demanded such a high price that I felt as though I was buying a new couch. On an obscure "personal computer" page of one of the video magazines, I spotted a product release for two desks available from O'Sullivan Industries at affordable prices. The photos showed enough flexible nooks and crannies to stash all of my ATARI® equipment and a shelf large enough to properly support the large portable television I use as a monitor.

No address or phone number was given in the release and it took several phone calls to the editorial offices of the magazine before I could locate O'Sullivan Industries. Once I had located the manufacturer, everything went *like clockwork*. Their helpful sales representative arranged for the model CT709 to be shipped out immediately.



MODEL CT709

Of the two available models, the CT709 is the larger and more permanent. It features a desktop large enough to comfortably accommodate most keyboards and printers; a removable, raised shelf to hold a monitor over the keyboard; and an open shelf and sliding drawer under the



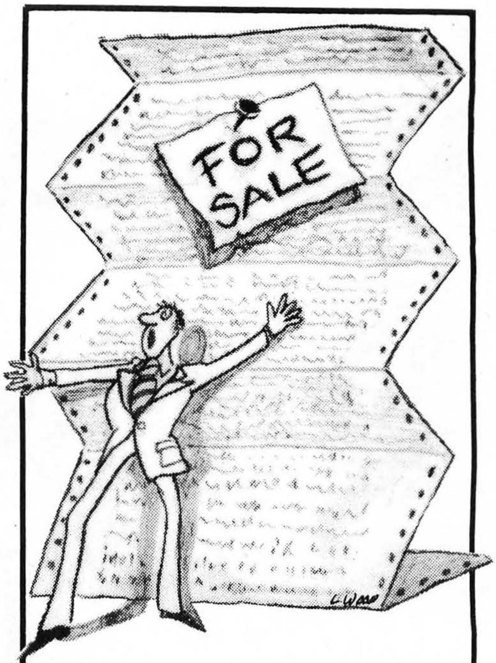
MODEL CT707

desktop to accommodate disk drives, modems, interfaces, joysticks, etc.. — all of those things which had been creeping out from the computer area into my living space. The CT707 has a smaller desktop and no lower shelf. It does, however, contain a large storage drawer and raised shelf capable of accommodating a monitor and a printer. The more portable of the two units, it is mounted on black, dual-wheel casters.

The day the large, heavy carton arrived, my heart jumped into my throat as its upacking yielded board upon board and several bags of dowels, screws, brackets, etc.. I had the sudden fear that I had taken on a project beyond my limited assembly skills. However, a quick reading of the instructions eased my mind. It was a simple assembly process I could conquer step by step. Within an hour, a friend and I had the desk standing proudly in the center of the room, ready to organize even the most unwieldy of microcomputer systems. Each piece of the desk was carefully numbered, diagrammed, and drilled in such a way that an assembly mistake would be nearly impossible. The illustrated instructions left nothing out. A smile grew on my face as I realized they had even allowed enough slack in the design of the metal brackets used to hold the major pieces together to allow me to adjust for slight imperfections of edge finishing or warpage in the boards. In short, the process was extremely simple and yielded an attractive piece of organizational furniture.

I heartily recommend these desks to those of you whose computer systems are threatening to take up more space than a new member of the family. The design of the units is such that they will accommodate Apple, ATARI® or (with shelf removed) TRS-80® equipment with no modification. With the upper shelf slipped out of its brackets, the unit looks very tempting as an office desk.

A loud "bravo" to O'Sullivan Industries for bringing aesthetics and organization to an otherwise ungainly mesh of electronic components.



Escape from the Dungeons of the Gods

by Ray Sato

Translations by Alex Lee and Rich Bouchard
Encryption modifications by Rich Bouchard, William Kubeck, and Alan J. Zett

Escape from the Dungeons of the Gods **is a fantasy/adventure game for a 16K TRS-80® Model I/III, ATARI® 400/800, or Apple (with Applesoft).**

Along with several other rebels, you have been captured by the secret police of the evil and cruel King Safuis II. They have imprisoned you in a dismal prison cell in the legendary Dungeon of the Gods, while the king's forces continue to wreak destruction on the populace outside. You must escape to aid in the rebellion against the king's cruel tyranny.

Legends say that somewhere within the dungeon lies the Chest of the Gods which holds the power to destroy the king and his forces. This and several other treasures are said to be guarded by powerful monsters which must be slain in order to gain access to the treasures. Take courage, and may the blessings of the gods go with you!

Playing Notes

The computer will always give you a brief description of where you are, what objects you can see, and what exits are visible. You move and act by typing in simple commands, generally consisting of a verb and a noun. If the computer tells you that there is a sword in the room, for example, you might want to type in the command "GET SWORD". At a later time, you might then be able to use it to "FIGHT MONSTER" or for some other purpose. If you no longer want to carry it, you can "DROP SWORD" whenever you please. Since the computer looks only at the first three letters of the verb and the last three letters of the noun, you may use abbreviations such as "FIG TER" (for "FIGHT MONSTER") if you desire. Movement is accomplished by entering a single letter rather than a two-word command: N, S, E, W, U, or D for north, south, east, west, up, or down. Typing the single word "INVENTORY" (or "INV") will display a list of what you are carrying.

Part of the challenge of any fantasy/adventure game such as this is to figure out what you are able to do in a particular situation. Therefore, you will not find a list of all the verbs which the computer can understand, or a list of all the objects that you may discover. You may find yourself frustrated by what seem to be dead ends, and end up getting killed in the process. This is all part of the adventure, and a test of your ingenuity and perseverance.

Program Notes

The most obvious feature of the program listing is that a great deal of it looks like a garbled mess. The BASIC keywords are all there in their usual form, but the string assignment statements and DATA lines contain incomprehensible garbage. This is because all the room descriptions, object names, monsters, and verbs have been encoded. This has been done to preserve the value of the game. Anyone who types an adventure program from a listing is bound to be disappointed in the game's playability, since the player has gained many clues about how the plot develops. So, even though the typing is made somewhat more difficult by the scrambled words, we feel that this is the only reasonable way of publishing adventure programs in listed form.

In next month's issue of ***SoftSide*** we will include a detailed write-up of the Machine Language encryption routines which were created to modify the normal program listings. The encryption method is a simple one, which results in leaving punctuation unmodified, and inverting the order of the letters of the alphabet. This simple inversion process has the advantage of using the same routine to decode the text as was used to encode it. In *Escape from the Dungeons of the Gods*, the user's input is encoded, the internal searches and comparisons are done in encoded form, and then the response is decoded and printed by the subroutine at line 5.

In order to offset the proofreading problems created by this approach, we have included an expanded SWAT Table for the three versions of this program. (See the article *SWAT* elsewhere in this issue.) Instead of the normal 12-line/500-byte SWAT parameters, we have used 5-line/200-byte parameters. This means that you must modify the first line of *SWAT* in order to generate a table to compare with ours. After merging *SWAT* in the normal way, but before running it, simply edit or retype line 32000, 60000, or 65000 (depending on the version), changing "NU=12: B=500" to "NU=5: B=200". This will provide an expanded SWAT Table, enabling you to more easily pinpoint typos.

Variables

A: Present room location.
A$: Present room description.
B$: Extended room description.
C$: Room contents.

D$: Room exits.
D: Down destination.
E: East destination.
I$(x): List of objects.
I(x): Object locations.
M: Monster attack indicator.
M(x): Hit points of monsters.
M$(x): List of monsters
N: North destination.
P0: Player's permanent hit points.
P1: Player's present hit points.
P2: Player's experience points.
P3: Number of items carried by player.
P4: Item status flag.
P5: Player's strength level.
P6: Item status flag.
P7: Restored hit point counter.
S0-S7: Item status flags.
T: Temporary program counter.
T$: Temporary string.
TI: Time used.
V$(x): List of verbs.
V1: Verb input code.
V1$: Verb input.
V2: Noun input code.
V2$: Noun input.

FIRST ISSUE FREE!

COMPUTER GAMING WORLD (Dept. S)
1919 E. Sycamore #203
Anaheim, CA 92805

Yes! Enter my subscription for a year of CGW and send my first issue free (a total of seven issues). U.S. - $11.00; Canada - $16.00; Foreign Air $25.00.

Please check one:

□ Check or money order enclosed

□ Master Card

□ Visa

Name ____________________

Address ____________________

City/State ____________ Zip ________

Name of Cardholder ____________________

MC#/VISA# ____________________

Exp. Date ____________________

Signature ____________________

Get your first issue of COMPUTER GAMING WORLD free when you pay for a year subscription! For only $11.00 (U.S.) you get over a year's worth of reviews, strategy and tactics, playing aids, scenarios, contests, interviews, industry news, and more. CGW keeps you on the cutting edge of the computer gaming hobby and industry. Whether you have an Apple II, Atari, TRS-80, or other popular microcomputer, you will find many hours of entertaining and informative reading in the pages of CGW. Clip out and mail the coupon today and get seven bi-monthly issues at 40% off the newstand price.

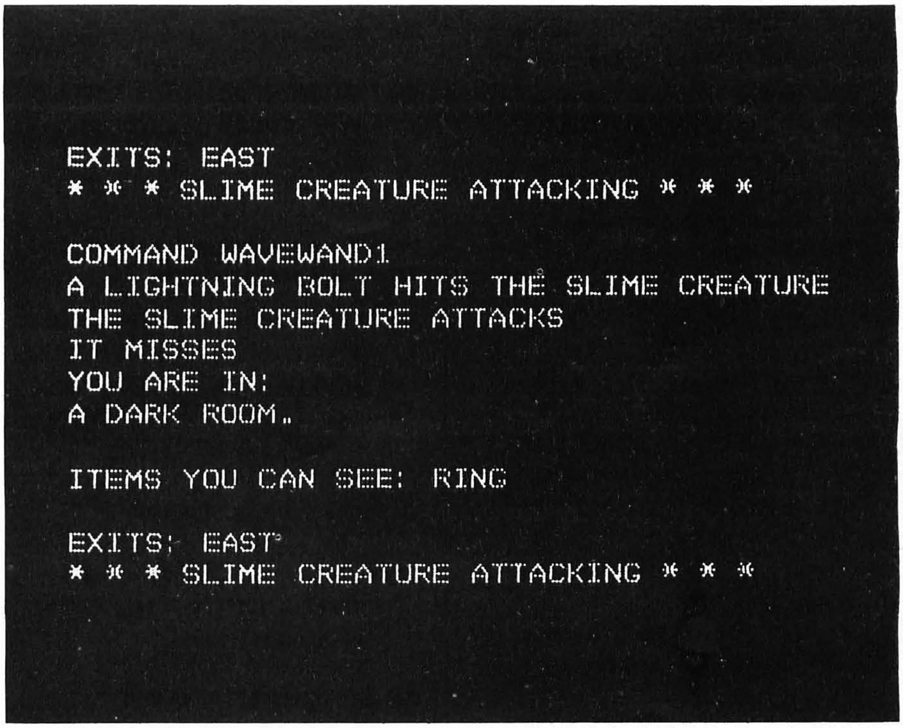
```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$   'DUNGEONS OF THE GODS'     $
$       AUTHOR: RAY SATO       $
$       TRANSL: ALEX LEE       $
$    (C) 1982    SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Jump to program initialization.

```
2  GOTO 2260
```

Decode and print output.

```
4  IF P$ = "" THEN  RETURN
5  FOR P = 1 TO  LEN (P$):II =  ASC
     ( MID$ (P$,P,N1)): PRINT  CHR$
     ( ABS ((C155 * (II > C64)) -
     II));: NEXT : PRINT : RETURN
```



```
EXITS: EAST
* * * SLIME CREATURE ATTACKING * * *

COMMAND WAVEWAND1
A LIGHTNING BOLT HITS THE SLIME CREATURE
THE SLIME CREATURE ATTACKS
IT MISSES
YOU ARE IN:
A DARK ROOM.

ITEMS YOU CAN SEE: RING

EXITS: EAST
* * * SLIME CREATURE ATTACKING * * *
```

Encode input.

```
6 V$ = "": IF VO$ = "" THEN  RETURN

7  FOR J = 1 TO  LEN (VO$):II =  ASC
     ( MID$ (VO$,J,N1)):V$ = V$ +
      CHR$ ( ABS ((C155 * (II > C
     64)) - II)): NEXT : RETURN
```

Descriptions of individual rooms.

```
10 A$ = "GSV DVHG VMW LU Z KIRHLM
      XVOO.  Z WLLI  OVZWH MLIGS"
     :B$ = "GSV WLLI RH OLXPVW":E
      = 2: GOTO 410
20 A$ = "GSV VZHG VMW LU Z KIRHLM
      XVOO.  Z NVGZO YLC ORVH LM
     GSV TILFMW":B$ = "GSV YLC RH
      OLXPVW":W = 1: GOTO 410
30 A$ = "Z GRMB KZHHZTV":N = 11:S
      = 4: GOTO 410
40 A$ = "Z GIRZMTFOZI ILLN":N = 3
     : GOTO 410
50 A$ = "Z HGLIZTV ILLN":N = 8:W =
     6: GOTO 410
60 A$ = "GSV DVZKLM ILLN":E = 5: GOTO
     410
70 A$ = "GSV GIVZHFIV EZFOG":N =
     12: GOTO 410
80 A$ = "Z HNZOO ILLN":S = 5:E =
     9: GOTO 410
90 A$ = "Z ILLN DRGS Z HGVVO WLLI
      RM GSV UOLLI":W = 8:E = 10:
     N = 13:S = 1: GOTO 410
 100 A$ = "GSV NFHRX SZOO":E = 11:
     W = 9: GOTO 410
110 A$ = "Z ILLN DRGS Z OZITV YIZ
     HH GZYOV":N = 14:S = 3:W = 1
     0: GOTO 410
120 A$ = "GSV GSILMV ILLN DRGS Z
     HGZGFV LM GSV    HLFGS DZOO"
     :N = 16:E = 13: GOTO 410
130 A$ = "Z ILLN DRGS Z HNZOO REL
     IB GZYOV":S = 9:W = 12:N = 1
     7:E = 14: GOTO 410
140 A$ = "GSV DRAZIW'H DLIPHSLK":
     S = 11:W = 13: GOTO 410
150 A$ = "Z ILLN DRGS HGZRIH TLRM
     T FK":E = 16:U = 36: GOTO 41
     0
160 A$ = "Z OZITV SZOO":S = 12:W =
     15:E = 17: GOTO 410
170 A$ = "Z OZITV HGLIZTV ILLN":S
      = 13:W = 16:N = 21:E = 18: GOTO
     410
180 A$ = "Z HNZOO ILLN DRGS HGZRI
     H TLRMT FK":W = 17:N = 22:U =
     39: GOTO 410
190 A$ = "GSV WIZTLM ILLN":E = 20
     : GOTO 410
200 A$ = "Z XLNKOVGVOB WZIP SZOO"
     :W = 19:E = 21: GOTO 410
210 A$ = "Z DLIPHSLK DRGS Z DLLWV
     M GZYOV":W = 20:E = 22:S = 1
     7: GOTO 410
220 A$ = "Z GRMB HJFZIV ILLN":N =
     25:S = 18:W = 21: GOTO 410
230 A$ = "Z IVXGZMTFOZI ILLN":E =
     24:N = 26: GOTO 410
240 A$ = "GSV PRGXSVM":W = 23:E =
     25: GOTO 410
250 A$ = "GSV WFMTVLM XZHRML DRGS
      Z OZITV HOLG    NZXSRMV":N =
     28:W = 24:S = 22: GOTO 410
 260 A$ = "GSV ORYIZIB":S = 23: GOTO
     410
270 A$ = "Z WZIP ILLN":E = 28: GOTO
     410
280 A$ = "GSV OZYLIZGLIB DRGS Z N
     VGZO GZYOV":S = 25:W = 27: GOTO
     410
290 A$ = "Z EVIB WRIGB ZMW WFHGB
     ILLN":S = 30: GOTO 410
300 A$ = "Z EVIB OLMT XLIIRWLI":N
      = 29:W = 31:S = 34: GOTO 41
     0
310 A$ = "Z YIRTSG XSZNYVI":E = 3
     0:W = 32: GOTO 410
320 A$ = "GSV XSZNYVI LU WVZGS":W
      = 33:E = 31: GOTO 410
330 A$ = "GSV XSZNYVI LU GSV TLWH
     ":E = 32: GOTO 410
340 A$ = "Z WZIP XLIIRWLI":N = 30
     :W = 35: GOTO 410
350 A$ = "Z HNZOO XSZNYVI DRGS Z
     HGVVO WLLI ZYLEV BLF":E = 34
     : GOTO 410
 360 A$ = "Z GRMB SZOO DRGS HGZRIH
      TLRMT WLDM":D = 15:E = 37: GOTO
     410
370 A$ = "Z OLMT, WZIP XLIIRWLI":
     S = 38:N = 40:W = 36:E = 39:
      GOTO 410
 380 A$ = "Z HGIZMTV, IVW ILLN. GS
     VIV RH Z YFGGLM  LM GSV MLIG
     S DZOO":N = 37: GOTO 410
```

```
390 A$ = "Z HNZOO ILLN DRGS HGZRI
     H TLRMT WLDM":D = 18:W = 37:
     GOTO 410
400 A$ = "Z SFTV VMGIZMXV SZOO":S
     = 37
```

Extended descriptions of current location.

```
410  IF A = 1 AND S0 = 1 THEN B$ =
     "GSV WLLI RH LKVM":N = 9
420  IF A = 2 AND S1 = 1 THEN B$ =
     "GSV YLC RH LKVM"
430  IF A = 35 AND S1 = 1 THEN B$
      = "Z HGZRIXZHV OVZWH FK GSI
     LFTS GSV HGVVO  WLLI":U = 9
440  IF A = 9 AND S2 = 1 THEN B$ =
     "GSVIV RH Z HGZRIXZHV OVZWRM
     T WLDM":D = 35
450  IF A = 25 AND S6 = 1 THEN B$
      = "GSVIV RH Z HGZRIXZHV WLD
     M":D = 29
460  IF A = 29 AND S6 = 1 THEN B$
      = "Z HGZRIXZHV VCGVMWH GSIL
     FTS GSV ILLU LU GSRH ILLN":U
      = 25
470  IF A = 12 AND S3 = 1 THEN B$
      = "GSVIV RH Z HVXIVG KZHHZT
     V OVZWRMT HLFGS":S = 7
480  IF A = 40 AND S4 = 0 THEN B$
      = "Z MLIGS WLLI RH OLXPVW"
490  IF A = 40 AND S4 = 1 THEN B$
      = "GSV MLIGS WLLI RH WVHGIL
     BVW":N = 50
```

Generate list of visible items and available exits for current location.

```
500 A$ = A$ + ".": FOR T = 1 TO 2
     1: IF A = I(T) THEN C$ = C$ +
     I$(T) + ", "
510  NEXT T: IF C$ <  > "" THEN C
     $ =  LEFT$ (C$, LEN (C$) - 2
     )
520  IF C$ = "" THEN C$ = "MLGSRM
     T"
530  IF N <  > 0 THEN D$ = D$ + "
     NORTH"
540  IF S <  > 0 THEN D$ = D$ + "
      SOUTH"
550  IF E <  > 0 THEN D$ = D$ + "
      EAST"
560  IF W <  > 0 THEN D$ = D$ + "
      WEST"
570  IF D <  > 0 THEN D$ = D$ + "
      DOWN"
580  IF U <  > 0 THEN D$ = D$ + "
      UP"
590  IF B$ <  > "" THEN B$ = B$ +
     "."
600  IF  LEFT$ (D$,1) = " " THEN
     D$ =  RIGHT$ (D$, LEN (D$) -
     1)
```

Describe current location, visible items, and available exits.

```
610  HOME : PRINT "YOU ARE IN:":P
     $ = A$: GOSUB 4:P$ = B$: GOSUB
     4: PRINT : PRINT "ITEMS YOU
     CAN SEE: ";:P$ = C$: GOSUB 4
     : PRINT : PRINT "EXITS: "D$
```

Describe monster, if present.

```
620  IF M <  > 0 THEN T$ = " ZGGZ
     XPRMT * * *":P$ = "* * * " +
     M$(M) + T$: GOSUB 5
```

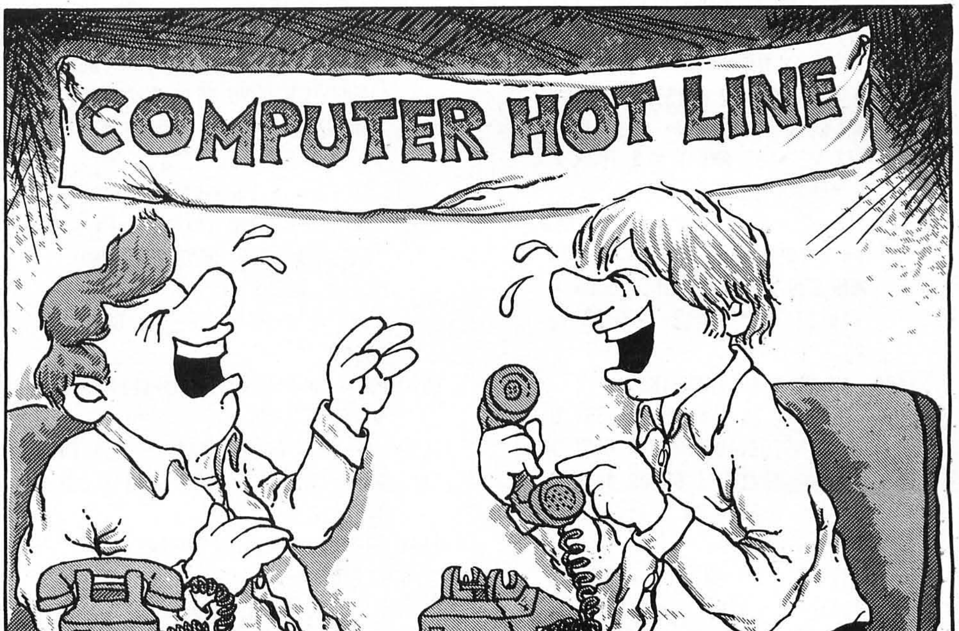
Get and interpret command.

```
630  PRINT : INPUT "COMMAND ";V0$
     : GOSUB 6
640  FOR T = 1 TO 7: IF V$ =  LEFT$
     (V$(T),1) THEN V$ = V$(T)
650  NEXT T
660  IF V$ = "FK" THEN V$ = "FK "

670  IF  LEN (V$) < 3 THEN 610
680 V1$ =  LEFT$ (V$,3):V2$ =  RIGHT$
     (V$,3): IF V2$ = "VHG" THEN
     V2$ = "LWH"
690  FOR T = 1 TO 23: IF V1$ =  LEFT$
     (V$(T),3) THEN V1 = T
700  NEXT : IF V1 = 0 THEN P$ = "
     R WLM'G FMWVIHGZMW BLF": GOSUB
     5: GOTO 1830
710  FOR T = 1 TO 21: IF V2$ =  RIGHT$
     (I$(T),3) THEN V2 = T
720  NEXT T
730  ON V1 GOTO 750,780,820,890,9
     20,940,980,1030,1050,1090,11
     90,1300,1830,1410,1520,1560,
     1600,1660,1690,1720,1760,180
     0,1820
740  GOTO 1830
```

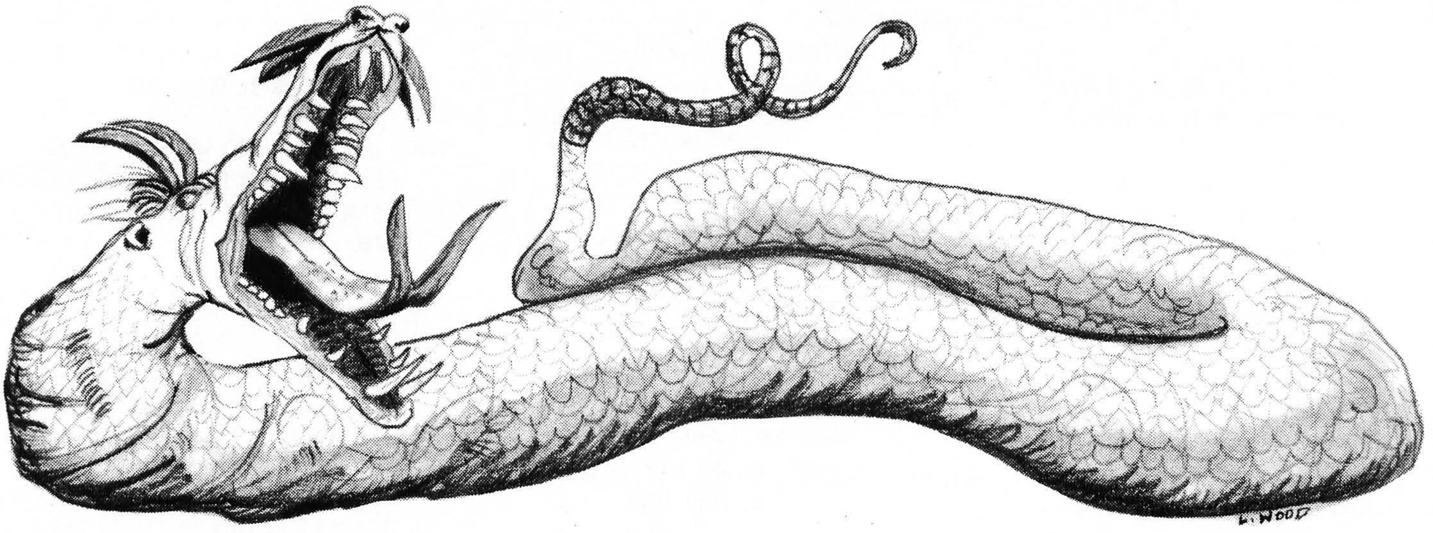
Command handler routines.

```
750  IF N = 0 THEN 970
760  IF A = 7 AND M > 0 THEN 960
770 A = N: PRINT "OK": GOTO 1830
780  IF S = 0 THEN 970
790 A = S: IF A = 7 AND M(1) > 0 THEN
     M = 1
800  PRINT "OK"
810  GOTO 1830
820  IF W = 0 THEN 970
830  IF M <  > 0 THEN 960
840 A = W: PRINT "OK": IF A = 19 AND
     M(3) > 0 THEN M = 3
850  IF A = 20 AND M(2) > 0 THEN
     M = 2
860  IF A = 27 AND M(4) > 0 THEN
     M = 4
870  IF A = 32 AND M(5) > 0 THEN
     M = 5
880  GOTO 1830
890  IF E = 0 THEN 970
900  IF M <  > 0 THEN 960
910 A = E: PRINT "OK": GOTO 1830
920  IF U = 0 THEN 970
930 A = U: GOTO 1830
940  IF D = 0 THEN 970
950 A = D: GOTO 1830
960 T$ = " YOLXPH GSV VCRG":P$ =
     "GSV " + M$(M) + T$: GOSUB 5
     : GOTO 1830
970 P$ = "GSVIV RH ML DZB GL TL G
     SZG WRIVXGRLM": GOSUB 4: GOTO
     1830
980  IF V2 = 2 AND A = 2 THEN P$ =
     "GSV YLC RH UZHGVMVW GL GSV
     UOLLI": GOSUB 4: GOTO 1830
990  IF V2 = 0 OR (I(V2) <  > A AND
     I(V2) <  > 0) THEN  PRINT "I
      DON'T SEE IT HERE": GOTO 18
     30
1000  IF I(V2) = 0 THEN  PRINT "Y
     OU ARE ALREADY CARRYING IT":
      GOTO 1830
1010  IF P3 = 10 THEN  PRINT "YOU
      CAN'T CARRY ANYTHING MORE":
      GOTO 1830
1020 I(V2) = 0:P3 = P3 + 1: PRINT
     "OK": GOTO 1830
1030  IF V2 = 0 OR I(V2) <  > 0 THEN
      PRINT "I DON'T HAVE IT": GOTO
     1830
1040 P3 = P3 - 1:I(V2) = A: PRINT
     "OK": GOTO 1830
1050  HOME : PRINT "PLAYER'S INVE
     NTORY"
1060  FOR T = 1 TO 21: IF I(T) =
     0 THEN P$ = I$(T): GOSUB 4
1070  NEXT
1080  GOTO 1920
1090  IF M = 0 THEN P$ = "GSVIV Z
     IVM'G ZMB NLMHGVIH SVIV": GOSUB
     4: GOTO 1830
1100  IF A = 20 AND P4 = 0 THEN P
     $ = "BLF XZM'G ZGGZXP RM GSV
      WZIP": GOSUB 4: GOTO 1830
1110  INPUT "WITH WHICH WEAPON? "
     ;V0$: GOSUB 6:V2$ = V$: IF  LEN
     (V2$) < 3 THEN 1110
```

```
1120 V2 = 0:V2$ =  RIGHT$ (V2$,3)
     : FOR T = 1 TO 21: IF V2$ =
      RIGHT$ (I$(T),3) THEN V2 =
     T
1130  NEXT : IF V2 = 0 OR I(V2) <
      > 0 THEN P$ = "BLF WLM'G SZ
     EV GSZG DVZKLM": GOSUB 4: GOTO
     1830
1134  IF V2 <  > 6 AND V2 <  > 14
      THEN P$ = "GSZG RHM'G Z DVZ
     KLM": GOSUB 4: GOTO 1830
1140  IF V2 = 14 AND M = 4 THEN P
     $ = "GSV UOZNRMT HDLIW PROOV
     W GSV HORNV XIVZ-GFIV": GOSUB
     4:P2 = P2 + 15:M = 0:M(4) =
     0: GOTO 1830
1150  IF  INT (100 *  RND (1)) +
     1 < (40 - P5) THEN P$ = "BLF
      NRHHVW GSV " + M$(M): GOSUB
     4: GOTO 1830
1160 P$ = "BLF SRG GSV " + M$(M):
      GOSUB 4:M(M) = M(M) - 5: IF
     M(M) <  = 0 THEN P$ = "RG RH
      WVZW": GOSUB 4:M(M) = 0:P2 =
     P2 + 15: IF M = 5 THEN P2 =
     P2 + 35
1170  IF M(M) = 0 THEN M = 0
1180  GOTO 1830
1190 T = V2: IF T = 0 OR I(T) <  >
     0 THEN P$ = "R WLM'G SZEV GS
     ZG": GOSUB 4: GOTO 1830
1200  IF T <  > 1 AND T <  > 12 AND
     T <  > 16 THEN P$ = "EVIB HG
     IZMTV. MLGSRMT SZKKVMH": GOSUB
     4: GOTO 1830
1210  IF T = 1 THEN P$ = "MLGSRMT
      SZKKVMH": GOSUB 4: GOTO 183
     0
1220  IF T = 12 AND M = 5 AND P6 <
     4 THEN P$ = "GSV YOZXP WIZTL
     M IVUOVXGH GSV ORTSGMRMT YLO
     G RMGL GSV DZOO": GOSUB 4:P6
      = P6 + 1: GOTO 1830
1230  IF T = 12 AND P6 > 2 THEN P
     $ = "MLGSRMT SZKKVMH": GOSUB
     4: GOTO 1830
1240  IF T = 12 AND M = 0 THEN P$
      = "Z ORTSGMRMT YLOG RH IVOV
     ZHVW YB GSV DZMW": GOSUB 4:P
     6 = P6 + 1: GOTO 1830
1250  IF T = 12 THEN P6 = P6 + 1:
     P$ = "Z ORTSGMRMT YLOG SRGH
     GSV " + M$(M): GOSUB 4:M(M) =
     M(M) - 15: IF M(M) <  = 0 THEN
     M(M) = 0:P2 = P2 + 15:M = 0:
     P$ = "RG RH WVZW": GOSUB 4
1260  IF T = 12 THEN 1830
1270  IF T = 16 AND A = 9 AND S2 =
     0 THEN S2 = 1:P$ = "Z HVXIVG
      HGZRIXZHV WLDM ZKKVZIH": GOSUB
     4: GOTO 1830

1280  IF T = 16 AND A = 12 AND S3
      = 0 THEN S3 = 1:P$ = "Z HVX
     IVG ILLN GL GSV HLFGS ZKKVZI
     H": GOSUB 4: GOTO 1830

1290 P$ = "MLGSRMT SZKKVMH": GOSUB
     4: GOTO 1830

1300  IF V2 = 2 AND I(1) = 0 AND
     S1 = 0 THEN S1 = 1:P$ = "GSV
      NVGZO YZI SVOKVW.  GSVIV RH
      Z PVB   RM GSV YLC": GOSUB
     4:I(3) = 2: GOTO 1830

1310  IF V2 = 2 AND I(1) = 0 AND
     S1 = 1 THEN P$ = "GSV YLC RH
      ZOIVZWB LKVM": GOSUB 4: GOTO
     1830

1320  IF V2$ = "LLI" AND A = 9 AND
     S2 = 1 THEN P$ = "GSV WLLI R
     H ZOIVZWB LKVM": GOSUB 4: GOTO
     1830

1330  IF V2$ = "LLI" AND A = 9 AND
     S2 = 0 THEN P$ = "GSV WLLI W
     LDM HVVNH NZTRXZOOB OLXPVW":
      GOSUB 4: GOTO 1830

1340  IF V2 = 21 AND (A = I(21) OR
     I(21) = 0) THEN P$ = "ZOO GS
     ZG BLF XZM HVV RH Z EVIB YIR
       TSG   ORTSG RMHRWV GSV XSVHG
       ": GOSUB 4: GOTO 1830

1350  IF V2$ = "LLI" AND S1 = 0 THEN
     P$ = "BLF ZIVM'G HGILMT VMLF
     TS GL LKVM RG": GOSUB 4: GOTO
     1830

1360  IF V2$ = "LLI" AND A = 1 AND
     I(3) <  > 0 THEN P$ = "BLF W
     LM'G SZEV GSV PVB": GOSUB 4:
      GOTO 1830

1370  IF V2$ = "LLI" AND A = 1 AND
     I(3) = 0 AND SO = 1 THEN P$ =
     "GSV WLLI RH ZOIVZWB LKVM": GOSUB
     4: GOTO 1830

1380  IF V2$ = "LLI" AND A = 1 AND
     I(3) = 0 AND SO = 0 THEN SO =
     1:P$ = "GSV WLLI RH MLD LKVM
     ": GOSUB 4: GOTO 1830

1390  IF V2$ = "LLI" AND A = 40 AND
     I(3) = 0 THEN P$ = "GSV PVB
     WLVHM'G URG": GOSUB 4: GOTO
     1830

1400 P$ = "R XZM'G WL GSZG": GOSUB
     4: GOTO 1830

1410  IF V2 = 0 OR I(V2) <  > 0 THEN
     P$ = "BLF WLM'G SZEV GSZG": GOSUB
     4: GOTO 1830
```



"HEY, HARRY! I'VE GOT ANOTHER ONE WHO'S BEEN TRYING TO FOLLOW THE INSTRUCTION MANUAL!"



```
1420  IF V2 = 17 AND A = 12 AND S
     3 = 0 THEN S3 = 1:P$ = "Z GI
     VZHFIV EZFOG RH VCKLHVW": GOSUB
     4:P3 = P3 - 1:I$(17) = "":I(
     17) =  - A: GOTO 1830
1430  IF V2 = 17 AND A = 9 AND S2
      = 0 THEN S2 = 1:P$ = "Z HGZ
     RIXZHV WLDM RH VCKLHVW": GOSUB
     4:P3 = P3 - 1:I$(17) = "":I(
     17) =  - A: GOTO 1830
1440  IF V2 = 17 AND A = 40 AND S
     4 = 0 THEN S4 = 1:P$ = "GSV
     MLIGS WLLI RH WVHGILBVW": GOSUB
     4:P3 = P3 - 1:I$(17) = "":I(
     17) =  - A: GOTO 1830
1450  IF V2 = 17 AND M <  > 0 THEN
     P$ = "GSV VCKOLHREV YZOO SRG
     H GSV " + M$(M): GOSUB 4:M(M
     ) = M(M) - 15:I$(17) = "":I(
     17) =  - A:P3 = P3 - 1: IF M
      <  = 0 THEN M(M) = 0:T = M:
     M = 0:P2 = P2 + 15:P$ = "RG
     RH WVZW": GOSUB 4: GOTO 1830
1460  IF V2 = 17 AND T = 5 THEN P
     2 = P2 + 35
1470  IF V2 = 17 AND M <  > 0 THEN
     P$ = "GSV YZOO VCKOLWVW ZTZR
     MHG GSV HGLMV DZOO": GOSUB 4
     :I$(17) = "":I(17) = 0:P3 =
     P3 - 1
1480  IF V2 = 17 THEN 1830
1490  IF V2 = 20 THEN P$ = "GSV U
     OZHP HSZGGVIVW RMGL NROORLMH
      LU    KRVXVH": GOSUB 4:I(20
     ) =  - A:P3 = P3 - 1:I$(20) =
     "": GOTO 1830
 1500  IF V2 = 13 AND M = 2 THEN I
      (13) =  - A:M(2) = 0:T$ = "
      SZH EZMRHSVW":P$ = "GSV " +
      M$(M) + T$: GOSUB 4:I$(13) =
      "":I(13) =  - A:M = 0:P2 = P
      2 + 15:P3 = P3 - 1: GOTO 183
      0
1510  GOTO 1030
1520  IF V2$ = "GFV" AND A = 12 AND
     S3 = 0 THEN S3 = 1:P$ = "Z H
     VXIVG KZHHZTV GL GSV HLFGS Z
     KKVZIH": GOSUB 4: GOTO 1830
1530  IF V2$ = "GFV" AND A = 12 AND
     S3 = 1 THEN S3 = 0:P$ = "GSV
      HLFGS KZHHZTV XOLHVW": GOSUB
     4: GOTO 1830
1540  IF V2$ = "GLM" AND A = 38 THEN
     A = 1:P$ = "BLF ZIV GVOVKLIG
     VW GL ....": GOSUB 4: GOTO 1
     830
1550 P$ = "R WLM'G FMWVIHGZMW DSZ
     G BLF DZMG NV GL  WL": GOSUB
     4: GOTO 1830
1560  IF V2 = 0 OR I(V2) <  > 0 THEN
     P$ = "R WLM'G SZEV GSZG": GOSUB
     4: GOTO 1830
1570  IF V2 = 4 THEN P5 = P5 -  INT
     (5 *  RND (1)) - 3:P$ = "BLF
      HFWWVMOB UVVO NFXS DVZPVI":
      GOSUB 4:I(4) =  - A:I$(4) =
     "":P3 = P3 - 1: GOTO 1830
1580  IF V2 = 9 THEN P5 = P5 + 5 +
      INT (5 *  RND (1)) + 1:P$ =
     "BLF HFWWVMOB UVVO NFXS HGIL
     MTVI": GOSUB 4:I(9) =  - A:I
     $(9) = "":P3 = P3 - 1: GOTO
     1830
1590 P$ = "WLM'G YV IRWRXFOLFH": GOSUB
     4: GOTO 1830
1600  IF V2 = 0 OR I(V2) <  > 0 THEN
     P$ = "R WLM'G SZEV GSZG": GOSUB
     4: GOTO 1830
1610  IF V2 = 4 THEN  PRINT "OK."
     :I$(4) = "":I(4) =  - A:P3 =
     P3 - 1: GOTO 1830
1620  IF V2 = 9 THEN  PRINT "OK."
     :I$(9) = "":I(9) =  - A:P3 =
     P3 - 1: GOTO 1830
1630  IF V2 = 20 AND S4 = 0 AND A
      = 40 THEN P$ = "GSV ZXRW WV
     HGILBVW GSV OLXPVW WLLI": GOSUB
     4:S4 = 1:I$(20) = "VNKGB UOZ
     HP":S5 = 1: GOTO 1830
1640  IF V2 = 20 AND S5 = 0 THEN
     P$ = "GSV ZXRW YFYYOVH RMGL
     GSV UOLLI": GOSUB 4:S5 = 1:I
     $(20) = "VNKGB UOZHP": GOTO
     1830
1650 P$ = "R WLM'G FMWVIHGZMW DSZ
     G BLF DZMG NV GL  WL": GOSUB
     4: GOTO 1830
1660  IF V2 = 0 OR I(V2) <  > 0 THEN
     P$ = "R WLM'G SZEV GSZG": GOSUB
     4: GOTO 1830
1670  IF V2 = 11 THEN I(V2) =  -
     A:I$(V2) = "":P3 = P3 - 1:S6
      = 1:P$ = "Z HGZRIXZHV WLDM
     RH VCKLHVW": GOSUB 4: GOTO 1
     830
1680 P$ = "R XZM'G WL GSZG": GOSUB
     4: GOTO 1830
1690  IF V2 <  > 10 THEN P$ = "R
     XZM'G WL GSZG": GOSUB 4: GOTO
     1830
1700  IF M = 1 THEN M(M) = 0:T$ =
     " RH XSZINVW":P$ = "GSV " +
     M$(M) + T$: GOSUB 4:M = 0:P2
      = P2 + 15: GOTO 1830
1710  PRINT "OK": GOTO 1830
1720  IF V2 <  > 18 THEN P$ = "R
     XZM'G IVZW GSZG": GOSUB 4: GOTO
     1830
1730  IF I(18) <  > 0 THEN P$ = "
     BLF WLM'G SZEV GSV HXILOO": GOSUB
     4: GOTO 1830
1740  IF M = 3 THEN T$ = " RH WVH
     GILBVW":P$ = "GSV " + M$(M) +
     T$: GOSUB 4:P3 = P3 - 1:I(18
     ) =  - A:I$(18) = "":M(M) =
     0:P2 = P2 + 15:M = 0: GOTO 1
     830
1750 P$ = "GSV HXILOO HZBH: IVZWR
     MT GSV NZTRX HKVOORMHXIRYVW
     LM GSRH SLOB WLXFNVMG DROO
      XZFHV GSV WVHGIFXGRLM LU Z
     MB IVW WIZTLM RM BLFI KIVHVM
     XV.": GOSUB 4: GOTO 1920
1760  IF V2 <  > 5 THEN  PRINT "W
     HAT?": GOTO 1830
1770  IF I(5) <  > 0 THEN P$ = "B
     LF WLM'G SZEV RG": GOSUB 4: GOTO
     1830
1780  IF P4 = 1 THEN P$ = "BLFI O
     ZNK RH ZOIVZWB LM": GOSUB 4:
      GOTO 1830
1790 P4 = 1:I$(5) = "ORG " + I$(5
     ): GOTO 1830
1800  HOME : PRINT "PLAYER'S STAT
     US:"
1810  PRINT "CURRENT HIT POINTS:"
     P1: PRINT "STRENGTH LEVEL:"P
     5: PRINT "EXP. POINTS:"P2: GOTO
     1920
1820  HOME : PRINT "GAME OVER":P2
      = P2 - 30: GOTO 2010
```

Update player status and conduct combat if appropriate.

```
1830 TI = TI + 1: IF P1 < P0 THEN
     P7 = P7 + .2: IF P7 = 1 THEN
     P7 = 0:P1 = P1 + 1
1840  IF M = 0 THEN 1940
1850  IF I(19) = 0 THEN 1940
1860 T =  INT (100 *  RND (1)) +
     1: IF T < 20 THEN 1940
1870 T$ = " ZGGZXPH":P$ = "GSV " +
     M$(M) + T$: GOSUB 4
1880 T =  INT (100 *  RND (1)) +
     1: IF T < 30 THEN  PRINT "IT
      MISSES": GOTO 1940
1890  PRINT "IT HITS YOU":P1 = P1
      - 10: IF I(7) = 0 THEN P1 =
     P1 + 5
1900  IF P1 < 0 THEN 2130
1910  GOTO 1940
1920  GET T$
1930  GOTO 1950
1940  FOR II = 1 TO 1500: NEXT
1950  FOR T = 0 TO 0
```

Initialize for new turn and jump to appropriate room description.

```
1960 GT = 1:N = 0:S = 0:E = 0:W =
     0:U = 0:D = 0:A$ = "":B$ = "
     ":C$ = "":D$ = "":V1 = 0:V2 =
     0:V$ = "":V1$ = "":V2$ = ""
1970  ON A GOTO 10,20,30,40,50,60
     ,70,80,90,100,110,120,130,14
     0,150,160,170,180,190,200,21
     0,220,230,240,250,260,270,28
     0,290,300,310,320,330,340,35
     0,360,370,380,390,400
1980 P$ = "BLF SZEV VHXZKVW UILN
     GSV WFMTVLM LU GSV TLWH!": GOSUB
     4
```

Calculate and display player ratings.

```
1990  PRINT "TOTAL ESCAPE TIME ="
     TI
2000  IF I(21) <  > 0 THEN P$ = "
     BLF WRWM'G TVG GSV XSVHG, GS
     V IVYVOORLM DROO YV XIFHSVW!
     !!": GOSUB 4: GOTO 2140
2010 P2 = P2 + 30: IF I(8) = 0 THEN
     P2 = P2 + 30
2020  IF I(15) = 0 THEN P2 = P2 +
     30
2030  IF I(21) = 0 THEN P2 = P2 +
     100
2040  PRINT "FINAL SCORE: "P2" OU
     T OF A POSSIBLE 300"
2050  PRINT "RANKING: ";
2060  IF P2 = 300 THEN  PRINT "EX
     PERT ADVENTURER": GOTO 2140
2070  IF P2 > 280 THEN  PRINT "CL
     ASS A EXPLORER": GOTO 2140
2080  IF P2 > 250 THEN  PRINT "CL
     ASS B EXPLORER": GOTO 2140
2090  IF P2 > 200 THEN  PRINT "CL
     ASS C EXPLORER": GOTO 2140
2100  IF P2 > 150 THEN  PRINT "FI
     RST CLASS SCOUT": GOTO 2140
2110  IF P2 > 100 THEN  PRINT "NO
     VICE SCOUT": GOTO 2140
2120  PRINT "BEGINNER": GOTO 2140

2130  PRINT "YOU ARE DEAD!!!": GOTO
      2140
```

"Play again?" routine. Restart or clean up and end.

```
2140  VTAB 22: CALL  - 958: PRINT
      "WOULD YOU LIKE TO TRY AGAIN
      ? (Y/N)";: GET IN$
2150  IF IN$ = "Y" THEN  GOTO 218
      0
2160  IF IN$ <  > "N" THEN 2140
2170  HOME : END
```

Initialize workspace. Read in items, verbs, and monsters.

```
2180  HOME : CLEAR : DIM I$(21),I
      (22),M(5),M$(5),V$(23)
```

```
2190  FOR T = 1 TO 21: READ I$(T)
      ,I(T): NEXT
2200  FOR T = 1 TO 23: READ V$(T)
      : NEXT
2210  FOR T = 1 TO 5: READ M$(T),
      M(T): NEXT
```

Establish player strength and hit points. Jump to first room.

```
2220  FOR T = 1 TO 15:P0 = P0 +  INT
      (2 * RND (1)) + 1: NEXT
2230 P1 = P0
2240  FOR T = 1 TO 5:P5 = P5 +  INT
      (5 * RND (1)) + 1: NEXT
2250 A = 1:N1 = 1:C155 = 155:C64 =
      64: GOTO 10
```

Display introduction and instructions.

```
2260  HOME : PRINT : PRINT  TAB(
      3)"ESCAPE FROM THE DUNGEON O
     F THE GODS"
2270  PRINT  TAB( 15)"BY RAY SATO
      ": PRINT  TAB( 6)"APPLE TRAN
      SLATION BY ALEX LEE"
2280  PRINT
2290  VTAB 7: PRINT "   FOR YEARS
      , THE SMALL MEDIEVAL TOWN
      OF ALVARD WAS RULED BY A KIN
      D MAN, KING SAFUIS I. HOWEVE
      R, WHEN HE DIED, HIS SONBECA
      ME THE NEW RULER. KING SAFUI
      S II WASA";
2300  PRINT " HARSH, CRUEL MAN WH
      O TURNED ALVARD IN-TO A CITY
        OF TERROR AND DESTRUCTION."
      : PRINT
2310  PRINT "   YOU AND SEVERAL O
      THER CITIZENS OF THETOWN HAD
       FORMED A SMALL ARMY TO REBE
      L   AGAINST THE EVIL RULER,
      BUT ON THE DAY  BEFORE YOUR
      FIRST ATTACK ON THE KING,
      SAFUIS' SECRET POLICE CAPTUR
      ED YOU";
2320  PRINT " AND  IMPRISONED YOU
       IN THE LEGENDARY DUNGEON OF
       THE GODS."
2330  PRINT : PRINT  TAB( 7)"PRES
      S ANY KEY TO CONTINUE";
2340  GET T$
2350  HOME : VTAB 5
2360  PRINT "   YOU ARE NOW BEIN
      G HELD IN A DISMAL  PRISON C
      ELL WHILE THE KING'S FORCES
      CON-TINUE TO DESTROY THE ENT
      IRE VILLAGE. YOUMUST ESCAPE
      FROM THE DUNGEON AND HELP
      THE REBELLION DEFEAT THE KIN
      G.";
2370  PRINT "  HERE IN THE DUNGEO
      N, LEGENDS SAY THERE IS A SP
      E-CIAL CHEST CALLED THE CHES
      T OF THE GODS,WHICH HAS THE
      POWER TO DESTROY THE KING AN
      D ALL HIS FORCES.  YOU MUST
      FIND THIS CHEST AND SEVERAL
      OTHER TREASURES";
2380  PRINT " HIDDENIN THE DUNGEO
      N. BUT BEWARE! SEVERAL POW-E
      RFUL MONSTERS PROTECT THE CO
      NTENTS OF  THIS DUNGEON AND
      YOU WILL BE FORCED TO  SLAY
      THEM IF YOU MEET THEM."
2390  PRINT : PRINT : PRINT  TAB(
      9)"PRESS ANY KEY TO BEGIN";:
       GET T$
2400  GOTO 2180
```

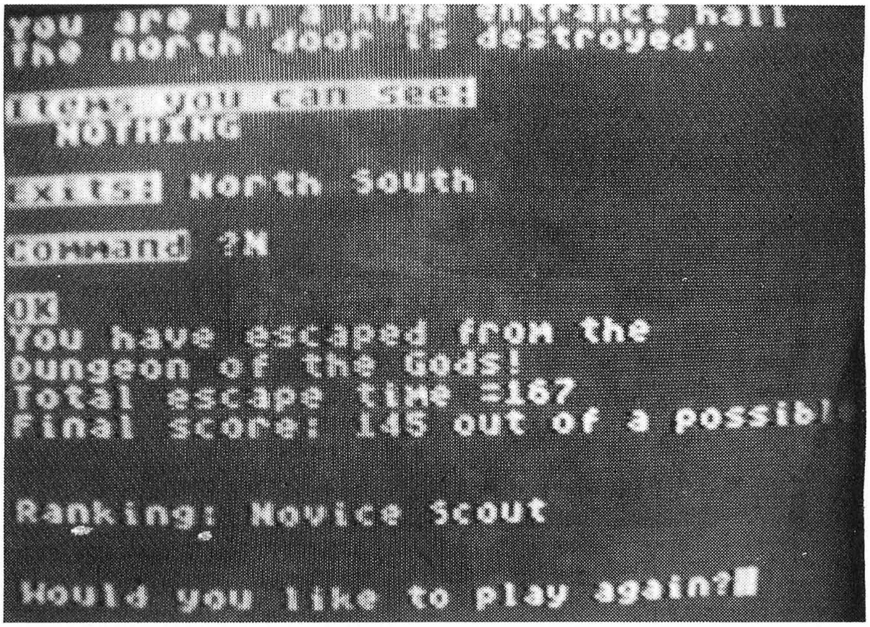
Item, Verb, and Monster data.

```
2410  DATA NVGZO YZI,2,NVGZO YLC,
      -2,PVB,-2,KLGRLM1,4,YIZHH OZ
      NK,5,OZMXV,6,HSRVOW,6,YZI LU
       KOZGRMFN,7,KLGRLM2,8,UOFGV,
      10,XLKKVI XLRM,11,DZMW1,13,N
      ZTRX WFHG,14,UOZNRMT HDLIW,1
      6,1WLAVM WRZNLMWH,19,DZMW2,2
      1,VCKOLHREV YZOO,23,HXILOO,2
      6,IRMT,27
2420  DATA UOZHP LU ZXRW,28,XSVHG
       LU GSV TLWH,33
2430  DATA MLIGS,HLFGS,DVHG,VZHG,
      FK ,WLDM,TVG,WILK,RMEVMGLIB,
      ZGGZXP,DZEV,LKVM,OLLP,GSILD,
      KFHS,WIRMP,KLFI,RMHVIG,KOZB,
      IVZW,ORTSG,HGZGFH,JFRG
2440  DATA XSILNZGRX HMZPV,15,YOZ
      XP XFYV,20,IVW WIZTLM,25,HOR
      NV XIVZGFIV,20,TRZMG YOZXP W
      IZTLM,35
```

APPLE SWAT TABLE FOR:
ESCAPE FROM THE DUNGEONS OF THE GODS
(Modified Parameters: NU = 5, B = 200)

LINES	SWAT CODE	LENGTH
2 - 7	ZE	177
10 - 30	TR	236
40 - 80	IT	180
90 - 120	MX	243
130 - 160	AJ	205
170 - 210	IH	260
220 - 250	LD	204
260 - 300	WO	217
310 - 350	BQ	232
360 - 390	FS	252
400 - 440	UX	237
450 - 480	RC	233
490 - 530	AP	180
540 - 580	QL	119
590 - 620	OY	201
630 - 670	RW	106
680 - 720	KL	171
730 - 770	SA	176
780 - 820	DA	70
830 - 870	WU	108
880 - 920	NO	70
930 - 970	VV	150
980 - 1010	NB	220
1020 - 1060	OW	168
1070 - 1110	LK	184
1120 - 1140	EV	259
1150 - 1190	PJ	244
1200 - 1220	OT	205
1230 - 1250	WS	242
1260 - 1300	TX	292
1310 - 1340	YK	304
1350 - 1370	US	204
1380 - 1410	LX	219
1420 - 1440	RZ	284
1450 - 1470	QZ	269
1480 - 1500	PP	233
1510 - 1540	MP	226
1550 - 1570	QF	209
1580 - 1610	WG	238
1620 - 1640	PL	237
1650 - 1680	LB	231
1690 - 1730	SI	230
1740 - 1750	TG	259
1760 - 1800	AF	179
1810 - 1850	SR	187
1860 - 1900	QB	175
1910 - 1950	IX	56
1960 - 1970	CS	245
1980 - 2010	GT	205
2020 - 2060	PB	149
2070 - 2110	WR	187
2120 - 2160	OH	147
2170 - 2210	TU	120
2220 - 2260	UF	156
2270 - 2290	OK	238
2300 - 2310	BY	288
2320 - 2360	KR	321
2370 - 2370	OE	212
2380 - 2390	JP	204
2400 - 2410	QW	244
2420 - 2440	JH	269

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          ATARI BASIC          $
$   "DUNGEONS OF THE GODS"      $
$        AUTHOR: RAY SATO       $
$     (C) 1982   SOFTSIDE       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```



```
You are in a huge entrance hall
the north door is destroyed.

Items you can see:
   NOTHING

Exits: North South

Command ?N

OK
You have escaped from the
Dungeon of the Gods!
Total escape time =167
Final score: 145 out of a possibl

Ranking: Novice Scout

Would you like to play again?
```

Initialize variables and jump to program initialization.

```
1 N0=0:N1=1:N2=2:N3=3:N4=4:N5=5:L1=970
:L2=1830:L3=960:L4=5030:DIM OK$(2):OK$
="OK"
2 GOTO 2260
```

Decode and print output.

```
4 IF P$="" THEN RETURN
5 FOR P=N1 TO LEN(P$):PRINT CHR$(ABS((
187*(P$(P,P)>"@"))-ASC(P$(P,P))));:NEX
T P:PRINT :RETURN
```

Encode input.

```
6 V$="":IF V0$="" THEN RETURN
7 FOR J=N1 TO LEN(V0$):V$(LEN(V$)+N1)=
CHR$(ABS((187*(V0$(J,J)>"@"))-ASC(V0$(
J,J)))):NEXT J:RETURN
```

Descriptions of individual rooms.

```
10 A$="GSV DVHG VMW LU Z KIRHLM   XVOO
.  z WLLI OVZWH MLIGS":B$="gSV WLLI RH
 OLXPVW":E=N2:RETURN
20 A$="GSV VZHG VMW LU Z KIRHLM   XVOO
. z NVGZO YLC ORVH LM GSV TILFMW":B$="
gSV YLC RH OLXPVW":W=N1:RETURN
30 A$="Z GRMB KZHHZTV":N=11:S=4:RETURN

40 A$="Z GIRZMTFOZI ILLN":N=N3:RETURN
50 A$="Z HGLIZTV ILLN":N=8:W=6:RETURN
60 A$="GSV DVZKLM ILLN":E=5:RETURN
70 A$="GSV GIVZHFIV EZFOG":N=12:RETURN

80 A$="Z HNZOO ILLN":S=5:E=9:RETURN
90 A$="Z ILLN DRGS Z HGVVO WLLI RMGSV
UOLLI":W=8:E=10:N=13:S=N1:RETURN
100 A$="GSV NFHRX SZOO":E=11:W=9:RETUR
N
110 A$="Z ILLN DRGS Z OZITV YIZHH  GZY
OV":N=14:S=N3:W=10:RETURN
120 A$="GSV GSILMV ILLN DRGS Z     HGZ
GFV LM GSV HLFGS DZOO":N=16:E=13:RETUR
N
130 A$="Z ILLN DRGS Z HNZOO RELIB  GZY
OV":S=9:W=12:N=17:E=14:RETURN
140 A$="GSV DRAZIW'H DLIPHSLK":S=11:W=
13:RETURN
150 A$="Z ILLN DRGS HGZRIH TLRMT   FK"
:E=16:U=36:RETURN
160 A$="Z OZITV SZOO":S=12:W=15:E=17:R
ETURN
170 A$="Z OZITV HGLIZTV ILLN":S=13:W=1
6:N=21:E=18:RETURN
180 A$="Z HNZOO ILLN DRGS HGZRIH   TLR
MT FK":W=17:N=22:U=39:RETURN
190 A$="GSV WIZTLM ILLN":E=20:RETURN
200 A$="Z XLNKOVGVOB WZIP SZOO":W=19:E
=21:RETURN
210 A$="Z DLIPHSLK DRGS Z DLLWVM   GZY
OV":W=20:E=22:S=17:RETURN
220 A$="Z GRMB HJFZIV ILLN":N=25:S=18:
W=21:RETURN
230 A$="Z IVXGZMTFOZI ILLN":E=24:N=26:
RETURN
240 A$="GSV PRGXSVM":W=23:E=25:RETURN
250 A$="GSV WFMTVLM XZHRML DRGS Z  OZI
TV HOLG NZXSRMV":N=28:W=24:S=22:RETURN
260 A$="GSV ORYIZIB":S=23:RETURN
270 A$="Z WZIP ILLN":E=28:RETURN
280 A$="GSV OZYLIZGLIB DRGS Z NVGZOGZY
OV":S=25:W=27:RETURN
290 A$="Z EVIB WRIGB ZMW WFHGB     ILL
N":S=30:RETURN
300 A$="Z EVIB OLMT XLIIRWLI":N=29:W=3
1:S=34:RETURN
310 A$="Z YIRTSG XSZNYVI":E=30:W=32:RE
TURN
320 A$="GSV XSZNYVI LU WVZGS":W=33:E=3
1:RETURN
330 A$="GSV XSZNYVI LU GSV TLWH":E=32:
RETURN
340 A$="Z WZIP XLIIRWLI":N=30:W=35:RET
URN
350 A$="Z HNZOO XSZNYVI DRGS Z     HGV
VO WLLI ZYLEV BLF":E=34:RETURN
360 A$="Z GRMB SZOO DRGS HGZRIH   TLR
MT WLDM":D=15:E=37:RETURN
370 A$="Z OLMT, WZIP XLIIRWLI":S=38:N=
40:W=36:E=39:RETURN
380 A$="Z HGIZMTV, IVW ILLN.  gSVIVRH
Z YFGGLM LM GSV MLIGS DZOO":N=37:RETUR
N
390 A$="Z HNZOO ILLN DRGS HGZRIH   TLR
MT WLDM":D=18:W=37:RETURN
400 A$="Z SFTV VMGIZMXV SZOO":S=37:RET
URN
```

Extended descriptions of current location.

```
410 IF A=N1 AND S0=N1 THEN B$="gSV WLL
I RH LKVM":N=9
420 IF A=N2 AND S1=N1 THEN B$="gSV YLC
 RH LKVM"
430 IF A=35 AND S1=N1 THEN B$="z HGZRI
XZHV OVZWH FK GSILFTS GSV HGVVOWLLI":U
=9
440 IF A=9 AND S2=N1 THEN B$="gSVIV RH
 Z HGZRIXZHV OVZWRMT WLDM":D=35
450 IF A=25 AND S6=N1 THEN B$="gSVIV R
H Z HGZRIXZHV WLDM":D=29
460 IF A=29 AND S6=N1 THEN B$="z HGZRI
XZHV VCGVMWH GSILFTS GSV ILLU  LU GSRH
 ILLN":U=25
470 IF A=12 AND S3=N1 THEN B$="gSVIV R
H Z HVXIVG KZHHZTV OVZWRMT     HLFGS":
S=7
480 IF A=40 AND S4=N0 THEN B$="z MLIGS
 WLLI RH OLXPVW"
490 IF A=40 AND S4=N1 THEN B$="gSV MLI
```

```
6S WLLI RH WVH6ILBVW":N=50
```

Generate list of visible items and available exits for current location.

```
500 REM
520 IF C$="" THEN C$="mlgsrmt"
530 IF N<>NO THEN D$(LEN(D$)+N1)="mLIG
S"
540 IF S<>NO THEN D$(LEN(D$)+N1)=" hLF
6S"
550 IF E<>NO THEN D$(LEN(D$)+N1)=" vZH
6"
560 IF W<>NO THEN D$(LEN(D$)+N1)=" dVH
6"
570 IF D<>NO THEN D$(LEN(D$)+N1)=" wLD
M"
580 IF U<>NO THEN D$(LEN(D$)+N1)=" fK"
590 IF B$<>"" THEN B$(LEN(B$)+N1)="."
600 IF D$<>"" THEN IF D$(N1,N1)=" " TH
EN D$=D$(N2,LEN(D$))
```

Describe current location, visible items, and available exits.

```
610 GRAPHICS NO:? "You are in ";:P$=A$
:GOSUB N4:P$=B$:GOSUB N4:PRINT :PRINT
"Items you can see: "
612 P$="  ":FOR T=N1 TO 21:IF A=I(T) T
HEN P$(N3)=I$(IP(T),IP(T+N1)-N1):GOSUB
 N4
614 NEXT T:IF P$="  " THEN P$="  mlgsr
mt":GOSUB N4
618 PRINT :PRINT "Exits: ";:P$=D$:GOSU
B N4
```

Describe monster, if present.

```
620 IF M<>NO THEN ? :P$="**** ":P$(6)=
M$(MP(M),MP(M+N1)-N1):P$(LEN(P$)+N1)="
 Z66ZXPRMT ****":GOSUB N5
```

Get and interpret command.

```
630 PRINT :PRINT "Command ";:INPUT VO$
:GOSUB 6
632 ?
640 FOR T=N1 TO 7:IF V$=VB$(T*N3-N2,T*
N3-N2) THEN V$=VB$(T*N3-N2,T*N3)
650 NEXT T
670 IF LEN(V$)<N3 THEN 610
680 V1$=V$(N1,N3):V2$=V$(LEN(V$)-2)
682 IF V2$="vhg" THEN V2$="lwh"
690 FOR T=N1 TO 23:IF V1$=VB$(T*N3-N2,
T*N3) THEN V1=T:GOTO 702
700 NEXT T
702 IF V1=NO THEN P$="r WLM'6 FMWVIH6Z
MW BLF":GOSUB N5:GOTO L2
710 FOR T=N1 TO 21:IF V2$=I$(IP(T+N1)-
N3,IP(T+N1)-N1) THEN V2=T:GOTO 730
720 NEXT T
730 ON V1 GOTO 750,780,820,890,920,940
,980,1030,1050,1090,1190,1300,1830,141
0,1520,1560,1600,1660,1690
732 ON V1-19 GOTO 1720,1760,1800,1820
740 GOTO 1830
```

Command handler routines.

```
750 IF N=NO THEN GOTO L1
760 IF A=7 AND M>NO THEN GOTO L3
770 A=N:? OK$:GOTO L2
780 IF S=NO THEN GOTO L1
790 A=S:IF A=7 AND M(N1)>NO THEN M=N1
800 PRINT OK$
810 GOTO L2
820 IF W=NO THEN GOTO L1
830 IF M<>NO THEN GOTO L3
840 A=W:PRINT OK$:IF A=19 AND M(N3)>NO
 THEN M=N3
850 IF A=20 AND M(N2)>NO THEN M=N2
860 IF A=27 AND M(N4)>NO THEN M=N4
870 IF A=32 AND M(N5)>NO THEN M=N5
880 GOTO L2
890 IF E=NO THEN GOTO L1
900 IF M<>NO THEN GOTO L3
910 A=E:PRINT OK$:GOTO L2
920 IF U=NO THEN GOTO L1
930 A=U:GOTO L2
940 IF D=NO THEN GOTO L1
950 A=D:GOTO L2
960 P$="gSV ":P$(5)=M$(MP(M),MP(M+1)-1
):P$(LEN(P$)+1)=" YOLXPH 6SV VCR6":GOS
UB N5:GOTO L2
970 P$="gSVIV RH ML DZB 6L TL 6SZ6 WRI
VX6RLM":GOSUB N4:GOTO L2
980 IF V2=N2 AND A=N2 THEN P$="gSV YLC
 RH UZH6VMVW 6L 6SV UOLLI":GOSUB N4:GO
TO L2
990 IF V2=NO OR (I(V2)<>A AND I(V2)<>N
O) THEN PRINT "I don't see it here":GO
TO L2
1000 IF I(V2)=NO THEN PRINT "You are a
lready carrying it":GOTO L2
1010 IF P3=10 THEN PRINT "You can't ca
rry anything more":GOTO L2
1020 I(V2)=NO:P3=P3+N1:PRINT OK$:GOTO
L2
1030 IF V2=NO OR I(V2)<>NO THEN PRINT
"I don't have it":GOTO L2
1040 P3=P3-N1:I(V2)=A:PRINT OK$:GOTO L
2
1050 GRAPHICS NO:PRINT "Player's Inven
tory"
1060 FOR T=N1 TO 21:IF I(T)=NO THEN P$
=I$(IP(T),IP(T+N1)-N1):GOSUB N4
1070 NEXT T
1080 GOTO 1920
1090 IF M=NO THEN P$="gSVIV ZIVM'6 ZMB
 NLMH6VIH SVIV":GOSUB N4:GOTO L2
1100 IF A=20 AND P4=NO THEN P$="bLF XZ
M'6 Z66ZXP RM 6SV WZIP":GOSUB N4:GOTO
L2
1110 PRINT "With which weapon";:INPUT
V2$:IF LEN(V2$)<N3 THEN 1110
1112 ?
1114 VO$=V2$:GOSUB 6:V2$=V$
1120 V2=NO:V2$=V2$(LEN(V2$)-N2):FOR T=
N1 TO 21:IF V2$=I$(IP(T+N1)-N3,IP(T+N1
)-N1) THEN V2=T:GOTO 1132
1130 NEXT T
1132 IF V2=NO OR I(V2)<>NO THEN P$="bL
F WLM'6 SZEV 6SZ6 DVZKLM":GOSUB N4:GOT
O L2
1134 IF V2<>6 AND V2<>14 THEN P$="gSZ6
 RHM'6 Z DVZKLM":GOSUB N4:GOTO L2
1140 IF V2<>14 OR M<>4 THEN 1150
1142 P$="gSV UOZNRMT HDLIW PROOVW 6SV
HORNV    XIVZ6FIV":GOSUB N4:P2=P2+15:M
=NO:M(N4)=NO:GOTO L2
1150 IF INT(100*RND(0))+1<(40-P5) THEN
 P$="bLF NRHHVW 6SV ":P$(LEN(P$)+N1)=M
$(MP(M),MP(M+N1)-N1):GOSUB N4:GOTO L2
1160 P$="bLF SR6 6SV ":P$(LEN(P$)+N1)=
M$(MP(M),MP(M+N1)-N1):GOSUB N4:M(M)=M(
M)-N5
1162 IF M(M)<=NO THEN P$="r6 RH WVZW!"
:GOSUB N4:M(M)=NO:P2=P2+15:IF M=N5 THE
N P2=P2+35
1170 IF M(M)=NO THEN M=NO
1180 GOTO L2
1190 T=V2:IF T=NO OR I(T)<>NO THEN P$=
"r WLM'6 SZEV 6SZ6":GOSUB N4:GOTO L2
1200 IF T<>N1 AND T<>12 AND T<>16 THEN
 P$="eVIB H6IZMTV.  mL6SRMT SZKKVMH":G
OSUB N4:GOTO L2
1210 IF T=N1 THEN P$="mL6SRMT SZKKVMH"
:GOSUB N4:GOTO L2
1220 IF T<>12 OR M<>N5 OR P6>=N4 THEN
1230
1222 P$="gSV YOZXP WIZTLM IVUOVX6H 6SV
        ORTS6MRMT YLO6 RM6L 6SV DZOO"
:GOSUB N4:P6=P6+N1:GOTO L2
1230 IF T=12 AND P6>2 THEN P$="mL6SRMT
 SZKKVMH":GOSUB N4:GOTO L2
1240 IF T=12 AND M=NO THEN P$="z ORTS6
MRMT YLO6 RH IVOVZHVW          YB 6SV
DZMW":GOSUB N4:P6=P6+N1:GOTO 1830
1250 IF T<>12 THEN 1270
1252 P6=P6+N1:P$="z ORTS6MRMT YLO6 SR6
 6SV ":GOSUB N4
1253 P$=M$(MP(M),MP(M+N1)-N1):GOSUB N4
:M(M)=M(M)-15
1254 IF M(M)<=0 THEN M(M)=0:P2=P2+15:M
=NO:P$="r6 RH WVZW":GOSUB N4
1260 GOTO L2
1270 IF T=16 AND A=9 AND S2=NO THEN S2
```

```
=1:P$="z HVXIVG HGZRIXZHV WLDM ZKKVZIH
":GOSUB N4:GOTO L2
1280 IF T=16 AND A=12 AND S3=N0 THEN S
3=N1:P$="z HVXIVG ILLN GL GSV HLFGS ZK
KVZIH":GOSUB N4:GOTO L2
1290 P$="■LGSRMT SZKKVMH":GOSUB N4:GOT
O L2
1300 IF V2<>N2 OR I(N1)<>N0 OR S1<>N0
THEN 1310
1302 S1=N1:P$="gSV NVGZO YZI SVOKVW.
gSVIV RH Z PVB RM GSV YLC":GOSUB N4:I(
N3)=N2:GOTO L2
1310 IF V2=N2 AND I(N1)=N0 AND S1=N1 T
HEN P$="gSV YLC RH ZOIVZWB LKVM":GOSUB
 N4:GOTO L2
1320 IF V2$="lli" AND A=9 AND S2=N1 TH
EN P$="gSV WLLI RH ZOIVZWB LKVM":GOSUB
 N4:GOTO L2
1330 IF V2$="lli" AND A=9 AND S2=N0 TH
EN P$="gSV WLLI WLDM HVVNH NZTRXZOOB O
LXPVW":GOSUB N4:GOTO L2
1340 IF V2<>21 OR (A<>I(21) AND I(21)<
>N0) THEN 1350
1342 P$="zOO GSZG BLF XZM HVV RH Z EVI
B YIRTSG ORTSG,RMHRWV GSV XSVHG":GOSUB
 N4:GOTO L2
1350 IF V2$="lli" AND S1=N0 THEN P$="b
LF ZIVM'G HGILMT VMLFTS GL LKVM RG":GO
SUB N4:GOTO L2
1360 IF V2$="lli" AND A=N1 AND I(N3)<>
N0 THEN P$="bLF WLM'G SZEV GSV PVB":GO
SUB N4:GOTO L2
1370 IF V2$="lli" AND A=N1 AND I(N3)=N
0 AND S0=N1 THEN P$="gSV WLLI RH ZOIVZ
WB LKVM":GOSUB N4:GOTO L2
1380 IF V2$="lli" AND A=N1 AND I(N3)=N
0 AND S0=N0 THEN S0=N1:P$="gSV WLLI RH
 MLD LKVM":GOSUB N4:GOTO L2
1390 IF V2$="lli" AND A=40 AND I(N3)=N
0 THEN P$="gSV PVB WLVHM'G URG":GOSUB
N4:GOTO L2
1400 P$="r XZM'G WL GSZG":GOSUB N4:GOT
O L2
1410 IF V2=N0 OR I(V2)<>N0 THEN P$="bL
F WLM'G SZEV GSZG":GOSUB N4:GOTO L2
1420 IF V2=17 AND A=12 AND S3=N0 THEN
S3=1:P$="z GIVZHFIV EZFOG RH VCKLHVW":
GOTO 5000
1430 IF V2=17 AND A=9 AND S2=N0 THEN S
2=N1:P$="z HGZRIXZHV WLDM RH VCKLHVW":
GOTO 5000
1440 IF V2=17 AND A=40 AND S4=N0 THEN
S4=N1:P$="gSV MLIGS WLLI RH WVHGILBVW"
:GOTO 5000
1450 IF V2<>17 OR M=N0 THEN 1460
1452 IF V2=17 AND M<>0 THEN P$="gSV VC
KOLHREV YLNY SRG GSV ":GOSUB N4
 1453 IF V2=17 AND M<>0 THEN P$=M$(MP(M
),MP(M+N1)-N1):GOSUB N4:M(M)=M(M)-15
1454 GOSUB 5020:I(17)=-A:P3=P3-N1:IF M
<=N0 THEN M(M)=N0:T=M:M=0:P2=P2+15:P$=
"rG RH WVZW":GOSUB N4:GOTO L2
1460 IF V2=17 AND T=55 THEN P2=P2+35
1470 IF V2=17 AND M=N0 THEN P$="gSV YZ
OO VCKOLWVW ZTZRMHG GSV HGLMV   DZOO":
GOSUB N4:GOSUB 5020:I(17)=N0:P3=P3-N1
1480 IF V2=17 THEN GOTO L2
1490 IF V2=20 THEN P$="gSV UOZHP HSZGG
VIVW RMGL NROORLMH LU  KRVXVH":GOSUB N
4:I(20)=-A:P3=P3-1:I9=20:GOSUB 5030
1492 IF V2=20 THEN GOTO L2
1500 IF V2<>13 OR M<>N2 THEN 1510
1502 I(13)=-A:M(N2)=N0:P$="gSV ":P$(N5
)=M$(MP(M),MP(M+N1)-N1):P$(LEN(P$)+1)=
" SZH EZMRHSVW":GOSUB N4
1504 I9=13:GOSUB 5030:I(13)=-A:M=N0:P2
=P2+15:P3=P3-N1:GOTO L2
1510 GOTO 1030
1520 IF V2$="gfv" AND A=12 AND S3=N0 T
HEN S3=N1:P$="z HVXIVG KZHHZTV GL GSV
HLFGS ZKKVZIH":GOSUB N4:GOTO L2
1530 IF V2$="gfv" AND A=12 AND S3=N1 T
HEN S3=N0:P$="gSV HLFGS KZHHZTV XOLHVW
":GOSUB N4:GOTO L2
1540 IF V2$="gl■" AND A=38 THEN A=N1:P
$="bLF ZIV GVOVKLIGVW GL ....":GOSUB N
4:GOTO L2
1550 P$="r WLM'G FMWVIHGZMW":GOSUB N4:
P$="DSZG BLF DZMG NV GL WL":GOSUB N4:G
OTO L2
1560 IF V2=N0 OR I(V2)<>0 THEN P$="r W
LM'G SZEV GSZG":GOSUB N4:GOTO L2
1570 IF V2=N4 THEN P5=P5-INT(5*RND(0))
-3:P$="bLF HFWWVMOB UVVO NFXS DVZPVI":
GOSUB N4:I(N4)=-A:I9=4:GOSUB 5030
1572 IF V2=N4 THEN P3=P3-N1:GOTO L2
1580 IF V2<>9 THEN 1590
1582 P5=P5+6+INT(RND(0)*5):P$="bLF HFW
WVMOB UVVO NFXS HGILMTVI":GOSUB N4:I(9
)=-A:I9=9:GOSUB L2:P3=P3-N1:GOTO L2
1590 P$="wLM'G YV IRWRXFOLFH":GOSUB N4
:GOTO L2
1600 IF V2=N0 OR I(V2)<>0 THEN P$="r W
LM'G SZEV GSZG":GOSUB N4:GOTO L2
1610 IF V2=N4 THEN PRINT OK$:I9=4:GOSU
B L4:I(4)=-A:P3=P3-N1:GOTO L2
1620 IF V2=9 THEN PRINT OK$:I9=9:GOSUB
 L4:I(9)=-A:P3=P3-N1:GOTO L2
1630 IF V2<>20 OR S4<>N0 THEN 1660
1632 IF A<>40 THEN 1640
1634 P$="gSV ZXRW WVHGILBVW GSV OLXPVW
 WLLI":GOSUB N4:S4=N1:I$(IP(20),IP(21)
-1)="vnkgb uozhp  ":S5=N1:GOTO L2
1640 P$="gSV ZXRW YFYYOVH RMGL GSV UOL
LI":GOSUB N4:S5=N1:I$(IP(20),IP(21)-1)
="vnkgb uozhp  ":GOTO L2
```

```
1650 P$="r WLM'G FMWVIHGZMW":GOSUB N4:
P$="DSZG BLF DZMG NV GL WL.":GOSUB N4:
GOTO L2
1660 IF V2=NO OR I(V2)<>NO THEN P$="r
WLM'G SZEV GSZG":GOSUB N4:GOTO L2
1670 IF V2=11 THEN I(V2)=-A:I9=V2:GOSU
B 5030:P3=P3-N1:S6=N1:P$="z HGZRIXZHV
WLDM RH VCKLHVW":GOSUB N4:GOTO L2
1680 P$="r XZM'G WL GSZG":GOSUB N4:GOT
O L2
1690 IF V2<>10 THEN P$="r XZM'G WL GSZ
G":GOSUB N4:GOTO L2
1700 IF M<>1 THEN 1710
1702 M(M)=NO:P$="gSV ":P$(N5)=M$(MP(M)
,MP(M+1)-1):P$(LEN(P$)+N1)=" RH XSZINV
W":GOSUB N4:M=NO:P2=P2+15:GOTO L2
1710 PRINT OK$:GOTO L2
1720 IF V2<>18 THEN P$="r XZM'G IVZW G
SZG":GOSUB N4:GOTO L2
1730 IF I(18)<>NO THEN P$="bLF WLM'G S
ZEV GSV HXILOO":GOSUB N4:GOTO L2
1740 IF M<>3 THEN 1750
1742 P$="gSV ivw wiztlm RH WVHGILBVW":
GOSUB N4:P3=P3-N1
1744 I9=18:GOSUB 5030:I(I9)=-A:M(M)=NO
:P2=P2+15:M=NO:GOTO L2
1750 P$="gSV HXILOO HZBH:  iVZWRMT GSV
 NZTRX   HKVOO RMHXIRYVW LM GSRH SLOB
WLXFNVMG":GOSUB N4
1752 P$="DROO XZFHV GSV WVHGIFXGRLM LU
 ZMB IVW WIZTLM RM BLFI KIVHVMXV.":GOS
UB N4:GOTO 1920
1760 IF V2<>N5 THEN PRINT "What?":GOTO
 L2
1770 IF I(N5)<>NO THEN P$="bLF WLM'G S
ZEV RG":GOSUB N4:GOTO L2
1780 IF P4=N1 THEN P$="bLFI OZNK RH ZO
IVZWB LM":GOSUB N4:GOTO L2
1790 P4=N1:I$(IP(N5),IP(N5))=" ":I$(IP
(N5)+1,IP(N5+1))=CHR$(30):GOTO L2
1800 GRAPHICS NO:PRINT "Player's Statu
s:"
1810 PRINT :PRINT "Current hit points:
";P1:PRINT "Strength level:";P5:PRINT
"Exp. points:";P2:GOTO 1920
1820 GRAPHICS NO:PRINT "Game over":P2=
P2-30:GOTO 2010
```

Update player status and conduct combat if appropriate.

```
1830 TI=TI+N1:IF P1<P0 THEN P7=P7+0.2:
IF P7=N1 THEN P7=NO:P1=P1+N1
1840 IF M=NO THEN 1940
1850 IF I(19)=NO THEN 1940
1860 T=INT(RND(0)*100)+1:IF T<20 THEN
1940
1870 ? :P$="gSV ":P$(N5)=M$(MP(M),MP(M
+1)-1):P$(LEN(P$)+1)=" ZGGZXPH":GOSUB
N4
1880 T=INT(RND(0)*100):IF T<30 THEN PR
INT "It misses":GOTO 1940
1890 PRINT "It hits you":P1=P1-10:IF I
(7)=NO THEN P1=P1+N5
1900 IF P1<NO THEN 2130
1910 GOTO 1940
1920 ? :? "Hit any key to continue..."
:POKE 764,255
1922 IF PEEK(764)=255 THEN 1922
1924 POKE 764,255
1930 GOTO 1950
1940 FOR II=N1 TO 160:NEXT II
1950 REM
```

Initialize for new turn and jump to appropriate room description.

```
1960 GT=N1:N=NO:S=NO:E=NO:W=NO:U=0:D=N
0:A$="":B$="":C$="":D$="":V1=NO:V2=NO:
V$="":V1$="":V2$=""
1970 IF A<=40 THEN GOSUB A*10:GOTO 410
1980 P$="bLF SZEV VHXZKVW UILN GSV":GO
SUB N4:P$="wFMTVLM LU GSV tLWH!":GOSUB
 N4
```

Calculate and display player ratings.

```
1990 PRINT "Total escape time =";TI
2000 IF I(21)<>NO THEN P$="bLF WRWM'G
TVG GSV XSVHG, GSV          IVYVOORLM D
ROO YV XIFHSVW!!!":GOSUB N4:GOTO 2140
2010 P2=P2+30:IF I(8)=NO THEN P2=P2+30
2020 IF I(15)=NO THEN P2=P2+30
2030 IF I(21)=NO THEN P2=P2+100
2040 PRINT "Final score: ";P2;" out of
 a possible 300"
2050 PRINT :PRINT "Ranking: ";
2060 IF P2=300 THEN PRINT "Expert Adve
nturer":GOTO 2140
2070 IF P2>280 THEN PRINT "Class A Exp
lorer":GOTO 2140
2080 IF P2>250 THEN PRINT "Class B Exp
lorer":GOTO 2140
2090 IF P2>200 THEN PRINT "Class C Exp
lorer":GOTO 2140
2100 IF P2>150 THEN PRINT "First Class
 Scout":GOTO 2140
2110 IF P2>100 THEN PRINT "Novice Scou
t":GOTO 2140
2120 PRINT "Beginner":GOTO 2140
2130 PRINT "You are dead!!!"
```

"Play again?" routine. Restart or clean up and end.

```
2140 PRINT :PRINT :PRINT "Would you li
ke to play again?";
2142 CLOSE #N1:OPEN #N1,N4,NO,"K:"
2144 GET #N1,IN
2150 IF IN=89 THEN RUN
2160 IF IN<>78 THEN 2140
2170 GRAPHICS NO:STOP
```

Initialize workspace. Read in items, verbs, and monsters.

```
2180 GRAPHICS 0:DIM I$(1000),I(22),M(5
),M$(1000),VB$(69),IP(23),MP(6)
2182 DIM Z$(50),A$(80),B$(60),C$(80),D
$(80),P$(80),VO$(20),V$(20),V1$(3),V2$
(17)
2184 FOR T=N1 TO N5:M(T)=NO:NEXT T
2190 I$="":FOR T=N1 TO 21:IP(T)=LEN(I$
)+N1:READ Z$:I$(LEN(I$)+N1)=Z$:READ Z:
I(T)=Z:NEXT T:IP(T)=LEN(I$)+N1
2192 I(NO)=NO
2200 READ VB$
2210 M$="":FOR T=N1 TO N5:MP(T)=LEN(M$
)+N1:READ Z$:M$(LEN(M$)+N1)=Z$:READ Z:
M(T)=Z:NEXT T:MP(T)=LEN(M$)+N1
```

Establish player strength and hit points. Jump to first room.

```
2220 FOR T=N1 TO 15:P0=P0+INT(RND(0)*2
)+1:NEXT T
2230 P1=P0
2240 FOR T=N1 TO N5:P5=P5+INT(N5*RND(N
0))+N1:NEXT T
2250 A=N1:N1=N1:GOSUB 10:GOTO 410
```

EXPLANATION OF ATARI® LINE LISTINGS

SoftSide uses the following conventions in representing unprintable characters in ATARI® line listings, unless otherwise noted:

Characters (including blank spaces) which are underlined should be typed in inverse video.

When graphics or control characters are to be included in a string (between quotation marks), that fact will be noted in a nearby REMark. In such cases, graphics characters are represented by the corresponding lower-case letter, and control characters are represented by the corresponding unshifted key symbol. For example: The lower-case letter s represents a graphics cross, entered by holding down the CTRL key and then pressing the S key. The symbol = represents a control-down-arrow, entered by first pressing and releasing the ESC key, then holding down the CTRL key and pressing the = key. (See Appendix F, and the back cover, of the *ATARI® BASIC Reference Manual.*)

The one regular exception to the above is that a clear-screen character (ESC CTRL- <) is represented in listings by a right-hand brace, which looks like this: }

A shifted = is represented in the listings by a vertical line with a small gap in it: ¦

Display introduction and instructions.

```
 2260 GRAPHICS NO:PRINT :PRINT " Escape
 from the Dungeon of the Gods":POSITIO
N 15,N2:PRINT "by Ray Sato"
2270 PRINT "  Atari translation by Ric
h Bouchard"
2280 PRINT
2290 ? :? :? "  For years, the small m
edieval town  of Alvard was ruled by a
 kind man,"
2292 ? "King Safuis I.  However, when
he died,his son became the new ruler.
 King"
2300 ? "Safuis II was a harsh, cruel m
an who  turned Alvard into a city of t
error   and destruction."
2310 ? "  You and several other citize
ns of   the town had formed a small ar
my to"
2312 ? "rebel against the evil ruler,
but on  the day before your first atta
ck on"
2314 ? "the king, Safuis' secret polic
e cap-  tured you and imprisoned you i
n the"
2320 ? "legendary Dungeon of the Gods.
"
2330 ? :? "      Press any key to cont
inue";
2340 OPEN #1,4,0,"K:":GET #1,T:CLOSE #
1
2350 GRAPHICS 0:? :? :? :?
2360 ? "  You are now being held in a
dismal  prison cell while the king's f
orces"
2362 ? "continue to destroy the entire
 vill-  age.  You must escape from the
 dungeon";
2364 ? "and help the rebellion defeat
the      king.  Here in the dungeon, le
gends"
2370 ? "say there is a special chest c
alled   the Chest of the Gods, which h
as the"
2372 ? "power to destroy the king and
all his forces.  You must find this ch
est and"
2380 ? "several other treasures hidden
 in the dungeon.  But beware!  Several
 power-"
2382 ? "ful monsters protect the conte
nts of  this dungeon and you will be f
orced"
2384 ? "to slay them if you meet them.
"
2390 ? :? "       Press any key to begi
n";
2392 OPEN #1,4,0,"K:":GET #1,T:CLOSE #
1
2400 GOTO 2180
```

Item, Verb, and Monster data.

```
2410 DATA nvgzo yzi,2,nvgzo ylc,-2,pvb
,-2,klgrlm1,4,fmorg yizhh oznk,5,ozmxv
,6,hsrvow,6,yzi lu kozgrmfn,7
2412 DATA klgrlm2,8,uofgv,10,xlkkvi xl
rm,11,dzmw1,13,nztrx wfhg,14,uoznrmt h
dliw,16
2414 DATA 1 wlavm wrznlmwh,19,dzmw2,21
,vckolhrev yzoo,23,hxiloo,26,irmt,27,u
ozhp lu zxrw,28,xsvhg lu gsv tlwh,33
2430 DATA mlihlfdvhvzhfk wldtvgwilrmez
ggdzelkvollgsikfhwirklfrmhkozivzorthgz
jfr
2440 DATA xsilnzgrx hmzpv,15,yozxp xfy
v,20,ivw wiztlm,25,hornv xivzgfiv,20,t
rzmg yozxp wiztlm,35
2900 GOTO 2900
5000 GOSUB N4:P3=P3-N1:GOSUB 5020:I(17
)=-A:GOTO L2
5020 I9=17
5030 FOR II=IP(I9) TO IP(I9+N1)-N1:I$(
II,II)=" ":NEXT II:RETURN
```

ATARI® SWAT TABLE FOR:
ESCAPE FROM THE DUNGEONS OF THE GODS
(Modified Parameters: NU = 5, B = 200)

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
1 - 5	XD	237	1340 - 1360	DC	257
6 - 20	JP	294	1370 - 1400	QH	233
30 - 70	IS	221	1410 - 1430	QQ	214
80 - 110	JN	266	1440 - 1453	GQ	239
120 - 140	QN	239	1454 - 1470	GZ	253
150 - 170	BB	208	1480 - 1502	UI	288
180 - 210	XA	264	1504 - 1530	NT	243
220 - 250	UL	264	1540 - 1570	RX	310
260 - 300	YJ	266	1572 - 1590	HJ	206
310 - 340	AB	210	1600 - 1630	UX	201
350 - 370	IZ	228	1632 - 1640	GY	244
380 - 410	OD	248	1650 - 1670	HB	215
420 - 450	HV	249	1680 - 1702	UB	214
460 - 480	OE	217	1710 - 1742	JC	188
490 - 540	OR	154	1744 - 1752	JZ	243
550 - 590	XG	140	1760 - 1800	QZ	216
600 - 614	MN	207	1810 - 1850	VV	222
618 - 640	NM	210	1860 - 1880	AH	208
650 - 690	EP	134	1890 - 1922	MV	173
700 - 730	JK	277	1924 - 1960	OK	173
732 - 770	TC	113	1970 - 2000	CN	241
780 - 820	ES	74	2010 - 2050	PB	179
830 - 870	YL	135	2060 - 2100	VY	232
880 - 920	YI	63	2110 - 2142	UZ	157
930 - 970	RO	194	2144 - 2180	KA	153
980 - 1010	IH	214	2182 - 2190	IR	226
1020 - 1060	DJ	177	2192 - 2230	AL	166
1070 - 1110	NA	183	2240 - 2280	EH	197
1112 - 1132	SE	173	2290 - 2300	UM	267
1134 - 1150	VE	291	2310 - 2314	AW	243
1160 - 1190	KH	225	2320 - 2360	EB	246
1200 - 1222	WK	236	2362 - 2370	HL	248
1230 - 1252	BG	222	2372 - 2382	LP	247
1253 - 1270	WT	216	2384 - 2410	DI	251
1280 - 1302	KB	226	2412 - 2430	RN	262
1310 - 1330	ZY	202	2440 - 5030	PK	206

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          TRS-80 BASIC         $
$   "DUNGEONS OF THE GODS"      $
$       AUTHOR: RAY SATO        $
$    (C) 1982   SOFTSIDE        $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Jump to program initialization.

```
2 GOTO 2140
```

Decode and print output.

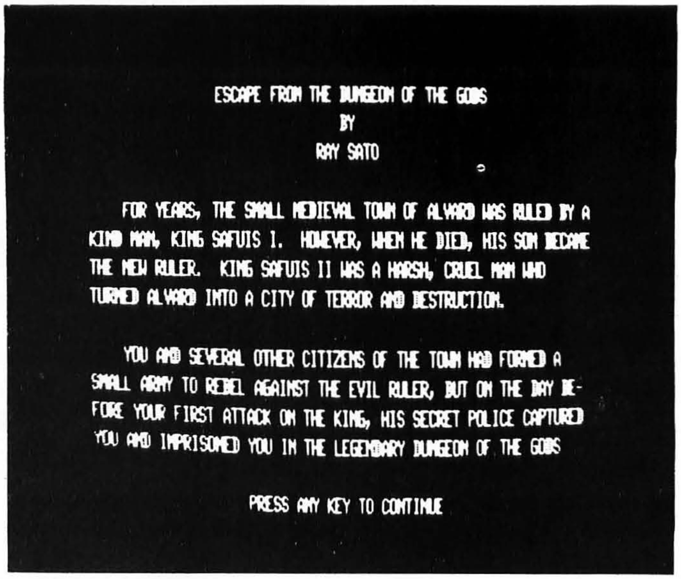
```
4 IFP$=""THENRETURNELSEFORP=1TOLEN(P$):II=ASC(MID$(P$,P,N1)):PRI
NTCHR$(ABS((C155*(II>C64))-II));:NEXT:PRINT:RETURN
5 GOTO2140
```

Encode input.

```
6 V$="":IFVO$=""THENRETURNELSEFORJ=1TOLEN(VO$):II=ASC(MID$(VO$,J
,N1)):V$=V$+CHR$(ABS((C155*(II>C64))-II)):NEXT:RETURN
```

Descriptions of individual rooms.

```
10 A$="GSV DVHG VMW LU Z KIRHLM XVOO.  Z WLLI OVZWH MLIGS":B$="G
SV WLLI RH OLXPVW":E=2:GOTO410:E=2:GOTO410
20 A$="GSV VZHG VMW LU Z KIRHLM XVOO.  Z NVGZO YLC ORVH LM GSV T
ILFMW":B$="GSV YLC RH OLXPVW":W=1:GOTO410
30 A$="Z GRMB KZHHZTV":N=11:S=4:GOTO410
40 A$="Z GIRZMTFOZI ILLN":N=3:GOTO410
50 A$="Z HGLIZTV ILLN":N=8:W=6:GOTO410
60 A$="GSV DVZKLM ILLN":E=5:GOTO410
70 A$="GSV GIVZHFIV EZFOG":N=12:GOTO410
80 A$="Z HNZOO ILLN":S=5:E=9:GOTO410
90 A$="Z ILLN DRGS Z HGVVO WLLI LM GSV UOLLI":W=8:E=10:N=13:S=1:
GOTO410
100 A$="GSV NFHRX SZOO":E=11:W=9:GOTO410
110 A$="Z ILLN DRGS Z OZITV YIZHH GZYOV":N=14:S=3:W=10:GOTO410
120 A$="GSV GSILMV ILLN DRGS Z HGZGFV LM GSV HLFGS DZOO":N=16:E=
13:GOTO410
130 A$="Z ILLN DRGS Z HNZOO RELIB GZYOV":S=9:W=12:N=17:E=14:GOTO
410
140 A$="GSV DRAZIW'H DLIPHSLK":S=11:W=13:GOTO410
150 A$="Z ILLN DRGS HGZRIH TLRMT FK":E=16:U=36:GOTO410
160 A$="Z OZITV SZOO":S=12:W=15:E=17:GOTO410
170 A$="Z OZITV HGLIZTV ILLN":S=13:W=16:N=21:E=18:GOTO410
180 A$="Z HNZOO ILLN DRGS HGZRIH TLRMT FK":W=17:N=22:U=39:GOTO41
0
190 A$="GSV WIZTLM ILLN":E=20:GOTO410
200 A$="Z XLNKOVGVOB WZIP SZOO":W=19:E=21:GOTO410
210 A$="Z DLIPHSLK DRGS Z DLLWVM GZYOV":W=20:E=22:S=17:GOTO410
220 A$="Z GRMB HJFZIV ILLN":N=25:S=18:W=21:GOTO410
230 A$="Z IVXGZMTFOZI ILLN":E=24:N=26:GOTO410
240 A$="GSV PRGXSVM":W=23:E=25:GOTO410
250 A$="GSV WFMTVLM XZHRML DRGS Z OZITV HOLG NZXSRMV":N=28:W=24:
S=22:GOTO410
260 A$="GSV ORYIZIB":S=23:GOTO410
270 A$="Z WZIP ILLN":E=28:GOTO410
280 A$="GSV OZYLIZGLIB DRGS Z NVGZO GZYOV":S=25:W=27:GOTO410
290 A$="Z EVIB WRIGB ZMW WFHGB ILLN":S=30:GOTO410
300 A$="Z EVIB OLMT XLIIRWLI":N=29:W=31:S=34:GOTO410
310 A$="Z YIRTSG XSZNYVI":E=30:W=32:GOTO410
```



ESCAPE FROM THE DUNGEON OF THE GODS
BY
RAY SATO

FOR YEARS, THE SMALL MEDIEVAL TOWN OF ALVARD WAS RULED BY A KIND MAN, KING SAFUIS I. HOWEVER, WHEN HE DIED, HIS SON BECAME THE NEW RULER. KING SAFUIS II WAS A HARSH, CRUEL MAN WHO TURNED ALVARD INTO A CITY OF TERROR AND DESTRUCTION.

YOU AND SEVERAL OTHER CITIZENS OF THE TOWN HAD FORMED A SMALL ARMY TO REBEL AGAINST THE EVIL RULER, BUT ON THE DAY BEFORE YOUR FIRST ATTACK ON THE KING, HIS SECRET POLICE CAPTURED YOU AND IMPRISONED YOU IN THE LEGENDARY DUNGEON OF THE GODS

PRESS ANY KEY TO CONTINUE

```
320 A$="GSV XSZNYVI LU WVZGS":W=33:E=31:GOTO410
330 A$="GSV XSZNYVI LU GSV TLWH":E=32:GOTO410
340 A$="Z WZIP XLIIRWLI":N=30:W=35:GOTO410
350 A$="Z HNZOO XSZNYVI DRGS Z HGVVO WLLI ZYLEV BLF":E=34:GOTO41
0
360 A$="Z GRMB SZOO DRGS HGZRIH TLRMT WLDM":D=15:E=37:GOTO410
370 A$="Z OLMT, WZIP XLIIRWLI":S=38:N=40:W=36:E=39:GOTO410
380 A$="Z HGIZMTV, IVW ILLN. GSVIV RH Z YFGGLM LM GSV MLIGS DZOO
":N=37:GOTO410
390 A$="Z HNZOO ILLN DRGS HGZRIH TLRMT WLDM":D=18:W=37:GOTO410
400 A$="Z SFTV VMGIZMXV SZOO":S=37
```

Extended descriptions of current location.

```
410 IFA=1ANDS0=1THENB$="GSV WLLI RH LKVM":N=9
420 IFA=2ANDS1=1THENB$="GSV YLC RH LKVM"
430 IFA=35ANDS2=1THENB$="Z HGZRIXZHV OVZWH FK GSILFTS GSV HGVVO
WLLI":U=9
440 IFA=9ANDS2=1THENB$="GSVIV RH Z HGZRIDZB OVZWRMT WLDM":D=35
450 IFA=25ANDS6=1THENB$="GSVIV RH Z HGZRIXZHV WLDM":D=29
460 IFA=29ANDS6=1THENB$="Z HGZRIXZHV VCGVMWH GSILFTS GSV ILLU LU
 GSRH ILLN":U=25
470 IFA=12ANDS3=1THENB$="GSVIV RH Z HVXIVG KZHHZTV OVZWRMT HLFGS
":S=7
480 IFA=40ANDS4=0THENB$="Z MLIGS WLLI RH OLXPVW"
490 IFA=40ANDS4=1THENB$="GSV MLIGS WLLI RH WVHGILBVW":N=50
```

Generate list of visible items and available exits for current location.

```
500 A$=A$+".":FORT=1TO21:IFA=I(T)THENC$=C$+I$(T)+", "
510 NEXTT:IFC$<>""THENC$=LEFT$(C$,LEN(C$)-2)
520 IFC$=""THENC$="MLGSRMT"
530 IFN<>0THEND$=D$+"MLIGS"
540 IFS<>0THEND$=D$+" HLFGS"
550 IFE<>0THEND$=D$+" VZHG"
560 IFW<>0THEND$=D$+" DVHG"
570 IFD<>0THEND$=D$+" WLDM"
580 IFU<>0THEND$=D$+" FK"
590 IFB$<>""THENB$=B$+"."
600 IFLEFT$(D$,1)=" "THEND$=RIGHT$(D$,LEN(D$)-1)
```

Describe current location, visible items, and available exits.

```
610 CLS:PRINT"YOU ARE IN :":P$=A$:GOSUB4:P$=B$:GOSUB4$:PRINT:PRI
NT"ITEMS YOU CAN SEE: ";:P$=C$:GOSUB4:PRINT:PRINT"EXITS: ";:P$=D
$:GOSUB4
```

Describe monster, if present.

```
620 IFM<>0THENP$="* * * "+M$(M)+" ZGGZXPRMT * * *":GOSUB4
```

Get and interpret command.

```
630 PRINT:INPUT"COMMAND";V0$:GOSUB6
640 FORT=1TO7:IFV$=LEFT$(V$(T),1)THENV$=V$(T)
650 NEXTT
660 IFV$="FK"THENV$="FK "
670 IFLEN(V$)<3THEN610
680 V1$=LEFT$(V$,3):V2$=RIGHT$(V$,3):IFV2$="VHG"THENV2$="LWH"
690 FORT=1TO23:IFV1$=LEFT$(V$(T),3)THENV1=T
700 NEXT:IFV1=0THENP$="R WLM'G FMWVIHGZMW BLF":GOSUB4:GOTO1830
710 FORT=1TO21:IFV2$=RIGHT$(I$(T),3)THENV2=T
720 NEXTT
730 ONV1GOTO750,780,820,890,920,940,980,1030,1050,1090,1190,1300
,1830,1410,1520,1560,1600,1660,1690,1720,1760,1800,1820
740 GOTO1830
```

Command handler routines.

```
750 IFN=0THEN970
760 IFA=7ANDM(1)>0THEN960
770 A=N:PRINT"OK":GOTO1830
780 IFS=0THEN970
790 A=S:IFA=7ANDM(1)>0THENM=1
800 PRINT"OK"
810 GOTO1830
820 IFW=0THEN970
830 IFM<>0THEN960
840 A=W:PRINT"OK":IFA=19ANDM(3)>0THENM=3
850 IFA=20ANDM(2)>0THENM=2
860 IFA=27ANDM(4)>0THENM=4
870 IFA=32ANDM(5)>0THENM=5
880 GOTO1830
890 IFE=0THEN970
900 IFM<>0THEN960
910 A=E:PRINT"OK":GOTO1830
920 IFU=0THEN970
930 A=U:GOTO1830
940 IFD=0THEN970
950 A=D:GOTO1830
960 P$="GSV "+M$(M)+" YOLXPH GSV VCRG":GOSUB4:GOTO1830
970 P$="GSVIV RH ML DZB GL TL RM GSZG WRIVXGRLM":GOSUB4:GOTO1830
980 IFV2=2ANDA=2THENP$="GSV YLC RH UZHGVMVW GL GSV UOLLI":GOSUB4
:GOTO1830
990 IFV2=0OR(I(V2)<>AANDI(V2)<>0)THENP$="R WLM'G HVV RG SVIV":GO
SUB4:GOTO1830
1000 IFI(V2)=0THENP$="BLF ZIV ZOIVZWB XZIIBRMT RG":GOSUB4:GOTO18
30
1010 IFP3=10THENP$="BLF XZM'G XZIIB ZMBGSRMT NLIV":GOSUB4:GOTO18
30
1020 I(V2)=0:P3=P3+1:PRINT"OK":GOTO1830
1030 IFV2=0ORI(V2)<>0THENP$="R WLM'G SZEV RG":GOSUB4:GOTO1830
1040 P3=P3-1:I(V2)=A:GOTO1830
1050 CLS:PRINT"PLAYER'S INVENTORY"
1060 FORT=1TO21:IFI(T)=0THENP$=I$(T):GOSUB4
1070 NEXT
1080 GOTO1920
1090 IFM=0THENP$="GSVIV ZIVM'G ZMB NLMHGVIH SVIV":GOSUB4:GOTO183
0
1100 IFA=20ANDP4=0THENP$="BLF XZM'G ZGGZXP RM GSV WZIP":GOSUB4:G
OTO1830
1110 INPUT"WITH WHICH WEAPON";V2$:IFLEN(V2$)<3THEN1110
1112 V0$=V2$:GOSUB6:V2$=V$
1120 V2=0:V2$=RIGHT$(V2$,3):FORT=1TO21:IFV2$=RIGHT$(I$(T),3)THEN
V2=T
1130 NEXT:IFV2=0ORI(V2)<>0THENP$="BLF WLM'G SZEV GSZG DVZKLM":GO
SUB4:GOTO1830
1134 IFV2<>6ANDV2<>14THENP$="GSZG RHM'G Z DVZKLM":GOSUB4:GOTO183
0
1140 IFV2=14ANDM=4THENP$="GSV UOZNRMT HDLIW PROOVW GSV HORNV XIV
ZGFIV":GOSUB4:P2=P2+15:M=0:M(4)=0:GOTO1830
1150 IFRND(100)<(40-P5)THENP$="BLF NRHHVW GSV "+M$(M):GOSUB4:GOT
O1830
1160 P$="BLF SRG GSV "+M$(M):GOSUB4:M(M)=M(M)-5:IFM(M)<=0THENP$=
"RG RH WVZW":GOSUB4:M(M)=0:P2=P2+15:IFM=5THENP2=P2+35
1170 IFM(M)=0THENM=0
1180 GOTO1830
1190 T=V2:IFT=0ORI(T)<>0THENP$="R WLM'G SZEV GSZG":GOSUB4:GOTO18
30
1200 IFT<>1ANDT<>12ANDT<>16THENP$="EVIB HGIZMTV. MLGSRMT SZKKVMH
":GOSUB4:GOTO1830
1210 IFT=1THENP$="MLGSRMT SZKKVMH":GOSUB4:GOTO1830
1220 IFT=12ANDM=5ANDS6<4THENP$="GSV YOZXP WIZTLM IVUOVXGH GSV OR
TSGMRMT YLOG RMGL GSV DZOO":GOSUB4:GOTO1830
1230 IFT=12ANDP6>2THENP$="MLGSRMT SZKKVMH":GOSUB4:GOTO1830
1240 IFT=12ANDM=0THENP$="Z YLOG LU ORTSGMRMT RH IVOVZHVW YB GSV
DZMW":GOSUB4:P6=P6+1:GOTO1830
1250 IFT=12THENP6=P6+1:P$="Z YLOG LU ORTSGMRMT SRGH GSV "+M$(M):
GOSUB4:M(M)=M(M)-15:IFM(M)<=0THENM(M)=0:P2=P2+15:M=0:P$="RG RH W
VZW":GOSUB4
1260 IFT=12THEN1830
1270 IFT=16ANDA=9ANDS2=0THENS2=1:P$="Z HVXIVG HGZRIXZHV WLDM ZKK
VZIH":GOSUB4:GOTO1830
1280 IFT=16ANDA=12ANDS3=0THENS3=1:P$="Z HVXIVG ILLN GL GSV HLFGS
 ZKKVZIH":GOSUB4:GOTO1830
1290 P$="MLGSRMT SZKKVMH":GOSUB4:GOTO1830
1300 IFV2=2ANDI(1)=0ANDS1=0THENS1=1:P$="GSV NVGZO YZI SVOKVW. GS
VIV RH Z PVB RMHRWV":GOSUB4:I(3)=2:GOTO1830
1310 IFV2=2ANDI(1)=0ANDS1=1THENP$="GSV YLC RH ZOIVZWB LKVM":GOSU
B4:GOTO1830
1320 IFV2$="LLI"ANDA=9ANDS2=1THENP$="GSV WLLI RH ZOIVZWB LKVM":G
OSUB4:GOTO1830
1330 IFV2$="LLI"ANDA=9ANDS2=0THENP$="GSV WLLI WLDM HVVNH GL YV O
LXPVW NZTRXZOOB":GOSUB4:GOTO1830
1340 IFV2=21AND(A=I(21)ORI(21)=0)THENP$="ZOO GSZG BLF XZM HVV RH
 Z EVIB YIRTSG ORTSG RMHRWV GSV XSVHG":GOSUB4:GOTO1830
1350 IFV2=2ANDS1=0THENP$="BLF ZIVM'G HGILMT VMLFTS GL LKVM RG":G
OSUB4:GOTO1830
1360 IFV2$="LLI"ANDA=1ANDI(3)<>0THENP$="BLF WLM'G SZEV GSV PVB":
GOSUB4:GOTO1830
1370 IFV2$="LLI"ANDA=1ANDI(3)=0ANDS0=1THENP$="GSV WLLI RH ZOIVZW
B LKVM":GOSUB4:GOTO1830
1380 IFV2$="LLI"ANDA=1ANDI(3)=0ANDS0=0THENS0=1:P$="GSV WLLI RH M
LD LKVM":GOSUB4:GOTO1830
1390 IFV2$="LLI"ANDA=40ANDI(3)=0THENP$="GSV PVB WLVHM'G URG":GOS
```

```
UB4:GOTO1830
1400 P$="R XZM'G WL GSZG":GOSUB4:GOTO1830
1410 IFV2=0ORI(V2)<>0THENP$="BLF WLM'G SZEV GSZG":GOSUB4:GOTO183
0
1420 IFV2=17ANDA=12ANDS3=0THENS3=1:P$="Z GIVZHFIV EZFOG RH VCKLH
VW":GOSUB4:P3=P3-1:I$(17)="":I(17)=-A:GOTO1830
1430 IFV2=17ANDA=9ANDS2=0THENS2=1:P$="Z HGZRIXZHV WLDM RH VCKLHV
W":GOSUB4:P3=P3-1:I$(17)="":I(17)=-A:GOTO1830
1440 IFV2=17ANDA=40ANDS4=0THENS4=1:P$="GSV MLIGS WLLI RH WVHGILB
VW":GOSUB4:P3=P3-1:I$(17)="":I(17)=-A:GOTO1830
1450 IFV2=17ANDM<>0THENP$="GSV VCKOLHREV YZOO SRGH GSV "+M$(M):G
OSUB4:M(M)=M(M)-15:I$(17)="":I(17)=-A:P3=P3-1:IFM<=0THENM(M)=0:T
=M:M=0:P2=P2+15:P$="RG RH WVZW":GOSUB4:GOTO1830
1460 IFV2=17ANDT=5THENP2=P2+35
1470 IFV2=17ANDM=0THENP$="GSV YZOO VCKOLWVW ZTZRMHG GSV HGLMV DZ
OO":GOSUB4:I$(17)="":I(17)=0:P3=P3-1
1480 IFV2=17THEN1830
1490 IFV2=20THENP$="GSV UOZHP HSZGGVIVW RMGL NROORLMH LU KRVXVH"
:GOSUB4:I(20)=-A:P3=P3-1:I$(20)="":GOTO1830
1500 IFV2=13ANDM=2THENI(13)=-A:M(2)=0:P$="GSV "+M$(M)+" SZH EZMR
HSVW":GOSUB4:I$(13)="":I(13)=-A:M=0:P2=P2+15:P3=P3-1:GOTO1830
1510 GOTO1030
1520 IFV2$="GFV"ANDA=12ANDS3=0THENS3=1:P$="Z HVXIVG KZHHZTV GL G
SV HLFGS ZKKVZIH":GOSUB4:GOTO1830
1530 IFV2$="GFV"ANDA=12ANDS3=1THENS3=0:P$="GSV HLFGS KZHHZTV XOL
HVW":GOSUB4:GOTO1830
1540 IFV2$="GLM"ANDA=38THENA=1:P$="BLF ZIV GVOVKLIGVW GL ....":G
OSUB4:GOTO1830
1550 P$="R WLM'G FMWVIHGZMW DSZG BLF DZMG NV GL WL":GOSUB4:GOTO1
830
1560 IFV2=0ORI(V2)<>0THENP$="R WLM'G SZEV GSZG":GOSUB4:GOTO1830
1570 IFV2=4THENP5=P5-(3+RND(5)):P$="BLF HFWWVMOB UVVO NFXS DVZPV
I":GOSUB4:I(4)=-A:I$(4)="":P3=P3-1:GOTO1830
1580 IFV2=9THENP5=P5+5+RND(5):P$="BLF HFWWVMOB UVVO NFXS HGILMTV
I':GOSUB4:I(9)=-A:I$(9)="":P3=P3-1:GOTO1830
1590 P$="WLM'G YV IRWRXFOLFH":GOSUB4:GOTO1830
1600 IFV2=0ORI(V2)<>0THENP$="R WLM'G SZEV GSZG":GOSUB4:GOTO1830
1610 IFV2=4THENPRINT"OK.":I$(4)="":I(4)=-A:P3=P3-1:GOTO1830
1620 IFV2=9THENPRINT"OK.":I$(9)="":I(9)=-A:P3=P3-1:GOTO1830
1630 IFV2=20ANDS4=0ANDA=40ANDS5=0THENP$="GSV ZXRW WVHGILBVW GSV
OLXPVW WLLI":GOSUB4:S4=1:I$(20)="VNKGB UOZHP":S5=1:GOTO1830
1640 IFV2=20ANDS5=0THENP$="GSV ZXRW YFYYOVH RMGL GSV UOLLI":GOSU
B4:S5=1:I$(20)="VNKGB UOZHP":GOTO1830
1650 P$="R WLM'G FMWVIHGZMW DSZG BLF DZMG NV GL WL":GOSUB4:GOTO1
830
1660 IFV2=0ORI(V2)<>0THENP$="R WLM'G SZEV GSZG":GOSUB4:GOTO1830
1670 IFV2=11THENI(V2)=-A:I$(V2)="":P3=P3-1:S6=1:P$="Z HGZRIXZHV
WLDM RH VCKLHVW":GOSUB4:GOTO1830
1680 P$="R XZM'G WL GSZG":GOSUB4:GOTO1830
1690 IFV2<>10THENP$="R XZM'G WL GSZG":GOSUB4:GOTO1830
1700 IFM=1THENM(M)=0:P$="GSV "+M$(M)+" RH XSZINVW":GOSUB4:M=0:P2
=P2+15:GOTO1830
1710 PRINT"OK":GOTO1830
1720 IFV2<>18THENP$="R XZM'G IVZW GSZG":GOSUB4:GOTO1830
1730 IFI(18)<>0THENP$="BLF WLM'G SZEV GSV HXILOO":GOSUB4:GOTO183
0
1740 IFM=3THENP$="GSV "+M$(M)+" RH WVHGILBVW":GOSUB4:P3=P3-1:I(1
8)=-A:I$(18)="":M(M)=0:P2=P2+15:M=0:GOTO1830
1750 P$="GSV HXILOO HZBH: IVZWRMT GSV NZTRX HKVOO RMHXIRYVW LM G
SRH SLOB WLXFNVMG DROO IVHFOG RM GSV WVHGIFXGRLM LU ZMB IVW WIZT
LM RM    BLFI KIVHVMXV.":GOSUB4:GOTO1920
1760 IFV2<>5THENPRINT"WHAT?":GOTO1830
1770 IFI(5)<>0THENP$="BLF WLM'G SZEV RG":GOSUB4:GOTO1830
1780 IFP4=1THENP$="BLFI OZNK RH ZOIVZWB LM":GOSUB4:GOTO1830
1790 P4=1:I$(5)="ORG "+I$(5):GOTO1830
1800 CLS:PRINT"PLAYER'S STATUS:"
1810 PRINT"CURRENT HIT POINTS:"P1:PRINT"STRENGTH LEVEL:"P5:PRINT
"EXP. POINTS:"P2:GOTO 1920
1820 CLS:PRINT"GAME OVER":P2=P2-30:GOTO2010
```

Update player status and conduct combat if appropriate.

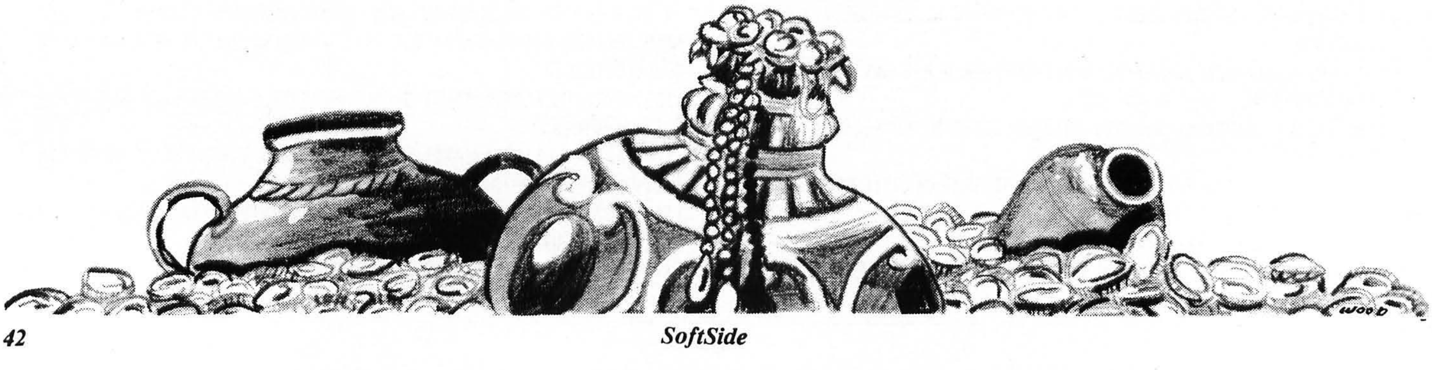
```
1830 TI=TI+1:IFP1<P0THENP7=P7+.2:IFP7=1THENP7=0:P1=P1+1
1840 IFM=0THEN1940
1850 IFI(19)=0THEN1940
1860 T=RND(100):IFT<20THEN1940
1870 P$="GSV "+M$(M)+" ZGGZXPH, ":GOSUB4
1880 T=RND(100):IFT<30THENP$="RG NRHHVH":GOSUB4:GOTO1940
1890 P$="RG SRGH BLF":GOSUB4:P1=P1-10:IFI(7)=0THENP1=P1+5
1900 IFP1<0THEN2130
1910 GOTO1940
1920 FORT=1TO700:IFINKEY$=""THENNEXT
1930 GOTO1950
1940 FORT=1TO150:IFINKEY$=""THENNEXT
1950 FORT=0TO0
```

Initialize for new turn and jump to appropriate room description.

```
1960 N=0:S=0:E=0:W=0:U=0:D=0:A$="":B$="":C$="":D$="":V1=0:V2=0:V
$="":V1$="":V2$=""
1970 ONAGOTO10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,1
60,170,180,190,200,210,220,230,240,250,260,270,280,290,300,310,3
20,330,340,350,360,370,380,390,400
1980 P$="BLF SZEV VHXZKVW UILN GSV WFMTVLM LU GSV TLWH!":GOSUB4
```

Calculate and display player ratings.

```
1990 PRINT"TOTAL ESCAPE TIME ="TI
2000 IFI(21)<>0THENP$="BLF WRWM'G TVG GSV XSVHG, GSV IVYVOORLM D
ROO YV XIFHSVW!!!":GOSUB4:GOTO2135
```



```
2010 P2=P2+30:IFI(8)=0THENP2=P2+30
2020 IFI(15)=0THENP2=P2+30
2030 IFI(21)=0THENP2=P2+100
2040 PRINT"FINAL SCORE: "P2" OUT OF A POSSIBLE 300"
2050 P$="IZMPRMT: ":GOSUB4
2060 IFP2=300THENP$="VCKVIG ZWEVMGFIVI":GOSUB4:GOTO 2135
2070 IFP2>280THENP$="XOZHH Z VCKOLIVI":GOSUB4:GOTO2135
2080 IFP2>250THENP$="XOZHH Y VCKOLIVI":GOSUB4:GOTO2135
2090 IFP2>200THENP"CLASS C EXPLORER":GOSUB4:GOTO2135
2100 IFP2>150THENP$="URIHG XOZHH HXLFG":GOSUB4:GOTO2135
2110 IFP2>100THENP"NOVICE SCOUT":GOSUB4:GOTO2135
2120 P$="YVTRMMVI":GOSUB4:GOTO2135
2130 P$="BLF ZIV WVZW!!!":GOSUB4
2135 INPUT "DO YOU WANT TO PLAY AGAIN? (Y/N)";IN$:IFLEFT$(IN$,1)
="Y"THENRUNELSECLS:END
```

"Play again?" routine. Restart or clean up and end.

```
2140 CLS:CLEAR800:DEFINTA-O,Q-Z:C155=-155:C64=64:P=0:II=0:N1=1:P
$="":DIMI$(21),I(22),M(5),M$(5),V$(23)
2150 GOSUB2240
2160 FORT=1TO21:READI$(T),I(T):NEXT
2170 FORT=1TO23:READV$(T):NEXT
```

Initialize workspace. Read in items, verbs, and monsters.

```
2180 FORT=1TO5:READM$(T),M(T):NEXT
2190 FORT=1TO15:PO=PO+RND(2):NEXT
2200 P1=PO
2210 FORT=1TO5:P5=P5+RND(5):NEXT
```

Establish player strength and hit points. Jump to first room.

```
2220 IFINKEY$=""THEN2220
2230 A=1:GOTO10
2240 CLS:PRINTTAB(15)"ESCAPE FROM THE DUNGEON OF THE GODS"
2250 PRINTTAB(31)"BY":PRINTTAB(28)"RAY SATO"
```

Display introduction and instructions.

```
2260 PRINT
2270 PRINT"    FOR YEARS, THE SMALL MEDIEVAL TOWN OF ALVARD WAS
RULED BY A"
2280 PRINT"KIND MAN, KING SAFUIS I.  HOWEVER, WHEN HE DIED, HIS
SON BECAME"
2290 PRINT"THE NEW RULER.  KING SAFUIS II WAS A HARSH, CRUEL MAN
 WHO"
2300 PRINT"TURNED ALVARD INTO A CITY OF TERROR AND DESTRUCTION."
2310 PRINT:PRINT"    YOU AND SEVERAL OTHER CITIZENS OF THE TOWN
HAD FORMED A "
2320 PRINT"SMALL ARMY TO REBEL AGAINST THE EVIL RULER, BUT ON TH
E DAY BE-"
2330 PRINT"FORE YOUR FIRST ATTACK ON THE KING, HIS SECRET POLICE
 CAPTURED"
2340 PRINT"YOU AND IMPRISONED YOU IN THE LEGENDARY DUNGEON OF TH
E GODS"
2360 PRINT:PRINTTAB(19)"PRESS ANY KEY TO CONTINUE";
2370 IFINKEY$=""THEN2370
2380 CLS
2390 PRINT"    YOU ARE NOW BEING HELD IN A DISMAL PRISON CELL WH
ILE THE"
2400 PRINT"KING'S FORCES CONTINUE TO DESTROY THE ENTIRE VILLAGE.
 YOU MUST"
2410 PRINT"ESCAPE FROM THE DUNGEON AND HELP THE REBELLION DEFEAT
 THE KING."
2420 PRINT"HERE IN THE DUNGEON, LEGENDS SAY THERE IS A SPECIAL C
HEST"
2430 PRINT"CALLED THE CHEST OF THE GODS, WHICH HAS THE POWER TO
DESTROY"
2440 PRINT"THE KING AND ALL HIS FORCES.  YOU MUST FIND THE CHEST
 AND SEV-"
2450 PRINT"ERAL OTHER TREASURES HIDDEN IN THE DUNGEON.  BUT BEWA
RE!!"
2460 PRINT"SEVERAL POWERFUL MONSTERS PROTECT THE CONTENTS OF THE
 DUNGEON"
2470 PRINT"AND YOU WILL BE FORCED TO SLAY THEM IF YOU MEET THEM.
"
2480 PRINT:PRINT:PRINTTAB(19)"PRESS ANY KEY TO BEGIN";:RETURN
```

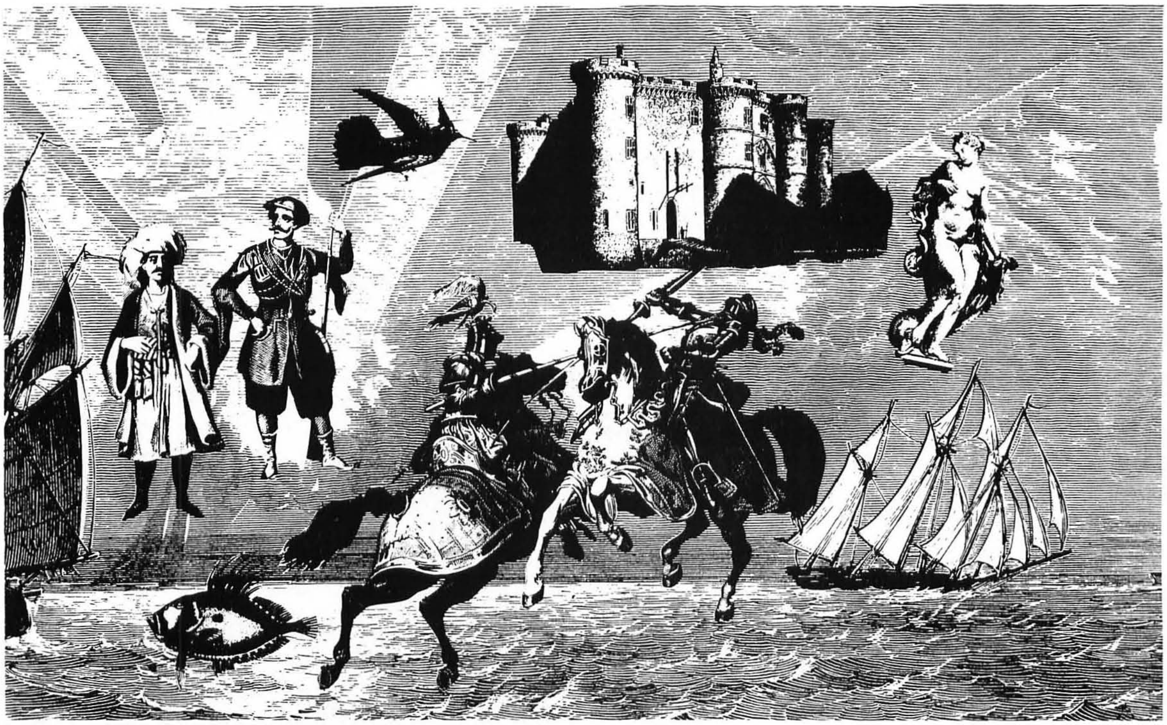
Item, Verb, and Monster data.

```
2490 DATANVGZO YZI,2,NVGZO YLC,-2,YIZHH PVB,-2,KLGRLM1,4,YIZHH O
ZNK,5,OZMXV,6,HSRVOW,6,YZI LU KOZGRMFN,7,KLGRLM2,8,UOFGV,10,XLKK
VI XLRM,11,DZMW1,13,NZTRX WFHG,14,UOZNRMT HDLIW,16,1 WLAVM WRZNL
MWH,19,DZMW2,21,VCKOLHREV YZOO,23,HXILOO,26,IRMT,27
2500 DATAUOZHP LU ZXRW,28,XSVHG LU GSV TLWH,33
2510 DATAMLIGS,HLFGS,DVHG,VZHG,FK ,WLDM,TVG,WILK,RMEVMGLIB,ZGGZX
P,DZEV,LKVM,OLLP,GSILD,KFHS,WIRMP,KLFI,RMHVIG,KOZB,IVZW,ORTSG,HG
ZGFH,JFRG
2520 DATAXSILNZGRX HMZPV,15,YOZXP XFYV,20,IVW WIZTLM,25,HORNV XI
VZGFIV,20,TRZMG YOZXP WIZTLM,35
```

TRS-80® SWAT TABLE FOR:
ESCAPE FROM THE DUNGEONS OF THE GODS
(Modified Parameters: NU = 5, B = 200)

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
2 - 10	ND	284	1340 - 1360	BA	233
20 - 50	UA	215	1370 - 1390	CV	202
60 - 100	GR	215	1400 - 1430	NJ	276
110 - 140	SB	243	1440 - 1450	WO	249
150 - 180	ZM	212	1460 - 1490	SE	226
190 - 230	HV	233	1500 - 1520	MY	204
240 - 280	DP	230	1530 - 1560	DX	245
290 - 330	EF	226	1570 - 1590	YZ	226
340 - 370	VQ	218	1600 - 1630	XT	253
380 - 410	LN	211	1640 - 1670	SF	284
420 - 450	EJ	211	1680 - 1720	VD	204
460 - 490	WZ	235	1730 - 1750	HG	312
500 - 540	QZ	152	1760 - 1800	RA	179
550 - 590	CT	116	1810 - 1850	OL	188
600 - 630	MM	207	1860 - 1900	PQ	167
640 - 680	ZQ	130	1910 - 1950	PC	73
690 - 730	LP	238	1960 - 1970	LQ	240
740 - 780	DX	76	1980 - 2010	CY	205
790 - 830	JZ	71	2020 - 2060	RM	159
840 - 880	WD	104	2070 - 2110	WL	208
890 - 930	BG	74	2120 - 2140	SL	210
940 - 980	VX	194	2150 - 2190	QJ	107
990 - 1020	TW	206	2200 - 2240	BQ	110
1030 - 1070	ZD	141	2250 - 2290	NF	243
1080 - 1112	SI	188	2300 - 2330	SN	270
1120 - 1140	YG	258	2340 - 2390	PH	196
1150 - 1190	CZ	237	2400 - 2420	VI	206
1200 - 1230	SO	243	2430 - 2450	HJ	203
1240 - 1250	RC	203	2460 - 2490	AJ	411
1260 - 1300	ZS	285	2500 - 2520	TC	269
1310 - 1330	TP	204			

The Adventure is



May 1982
Titanic

You are the Captain of the Titanic on her maiden voyage. Suddenly a large white object comes into view through the window. Can you avoid the historic collision? If not, can you save the lives of your passengers and crew?

April 1982
Witches' Brew Adventure

You find yourself in an enchanted forest. You must find your way to the castle and rescue the Princess who is chained inside its dungeon. A tightly-woven blend of fantasy, horror, and science fiction, this complex adventure will challenge your wits and ingenuity.

March 1982
James Brand Adventure

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

February 1982
Klondike Adventure

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

January 1982
Windsloe Mansion Adventure

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

December 1981
Black Hole Adventure

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

November 1981
Around the World in Eighty Days Adventure

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

October 1981
Crime Adventure

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

September 1981
Jack The Ripper Adventure

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

August 1981
Treasure Island Adventure

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

July 1981
Alien Adventure

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

June 1981
Arabian Adventure

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

Waiting for You...

JUNE ADVENTURE OF THE MONTH
ARROW ONE

You are Adam Trent, a trouble-shooter for the Federation of Space. You descend to an alien planet and make a horrifying discovery, which impels you into a desperate and dangerous quest. This unique science fiction adventure will test your skills and ingenuity.

Subscribe to Adventure of the Month

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

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

The cost? A six-month membership is just $29 for the tape ($4.83 per adventure) or $49 for the disk ($8.16 per adventure). If you're not sure that you can take six full months of excitement, you can order a single tape for $7 or a disk for $10. Or, if you're especially adventuresome, we're offering disks, packed with three great adventures, for only $26 per disk.

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

Adventure of the Month
6 South Street, Milford NH 03055

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

■ **Six month subscription:**
□ Cassette — $29
□ Disk — $49

■ **Individual adventures (please specify)**

____

____

____

□ Cassette — $7 each
□ Disk — $10 each

■ **Three adventures on one super disk ($26 each):**
□ Arabian, Alien, and Treasure Island Adventures
□ Jack the Ripper, Crime, and Around the World Adventures
□ Black Hole, Windsloe Mansion, and Klondike Adventures
□ James Brand, Witche's Brew, & Titantic Adventures

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

Name ________________________
Address ________________________
City/State ________________ Zip ________
□ Payment enclosed

□ MasterCard □ VISA Name of Cardholder ____________
MC# and Interbank#/VISA# ____________
Exp. Date ________ Signature ____________

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

Strategic Weapon

By Jon Voskuil
TRS-80® and ATARI® versions By Alan J. Zett

***SWAT* is a debugging utility program for any Apple with Applesoft, ATARI®, or TRS-80®.**

One of the major frustrations of typing in computer programs from printed listings is finding typographical errors. Every day the programming staff at ***SoftSide*** receives a number of letters and phone calls from readers who are in the midst of rediscovering one of Murphy's many laws, "There is always one more bug." Only a very few of these inquiries turn up an actual mistake in the published listing; the great majority result from hard-to-find typing errors. We are often able to help locate these errors during a phone call, if the program's behavior is distinctive enough or if one of us is familiar enough with the program's internal operation. But the process can be very time-consuming, for you and for us. We constantly wish that there were a better way to pin down the location of hidden typos in program lines.

Enter *SWAT*, our "Strategic Weapon Against Typos." Inspired by a program with a similar purpose in a recent issue of *Nibble* (to give credit where it's due), we've developed these three short programs to help you find differences between any program listing which we print and the lines which you type into your computer.

If you look at the programs in this issue, you will notice, at the end of every listing, a "*SWAT* Table" containing various numbers and letter codes. Such tables are the end result of running the *SWAT* program. The idea is that a *SWAT* Table created by us here at ***SoftSide***, using the program from which the magazine listing was made, should be identical to a *SWAT* Table created by you on your computer, using the program you have typed in from the listing. If there are any differences, you will know that there is a disagreement between the program in your computer's memory and the one in ours; and the *SWAT* Table will tell you, within a few program lines, where to find the error. (On those very rare occasions when the published listing goes astray from what's in OUR computer, it will help pinpoint that as well.)

What *SWAT* Does

Three different columns of information are generated by *SWAT*. Each entry in the first column is a range of line numbers; each entry in the second column is a two-letter "*SWAT* Code"; and each entry in the third column is the length, in bytes, of the specified program lines.

SWAT's operation is quite simple. Starting at the first memory location where program lines are stored, it begins examining the computer's memory. It finds the line number of the first line of the program, and the pointers which give the location (address) of the beginning of the next line. Jumping to that address, it then finds the address of the following line. In this way it scans twelve program lines or about 500 bytes of memory, whichever comes first. This information is incorporated into the three-column *SWAT* Table: The first entry in column one lists a range of line numbers, and the corresponding entry in column three gives the number of bytes of memory occupied by those lines. This procedure is repeated until the end of the program is reached.

This information alone would be quite valuable in finding typing errors in a program, but not all errors would be caught. Although extra or omitted characters would usually show up in the byte length number, simple mistyped characters would slip by unnoticed. So *SWAT* performs one additional calculation (the one which takes most of the time) as it goes through each group of program lines. It adds up the value of every byte (the numerical contents of every memory location) from the beginning to the end

Against Typos

of each group of lines. The resulting many-digit total is then converted into a base-26 number so that it can be represented by letters of the alphabet, and the rightmost two "digits" (letters) of this number become the *SWAT* Code in column two of the table.

How To Use *SWAT*

The first step, of course, is to type in the *SWAT* program and save it to tape or disk. (It can be used to check itself for typos, with the simple modifications described below.) Then, to use *SWAT* on another program, you must first append it to the end of the program you want to trouble-shoot. The most straightforward way of doing this is to LOAD it into memory before you even start typing in a program from a magazine listing. If you neglect to do this, however, it's not difficult to append it later, using the following procedures:

APPLE: If you have a disk, write the *SWAT* program to a disk text file; it can then be EXECed into memory at any time. If you don't have a disk, you can append *SWAT* from tape with the following three steps:

```
(a) POKE103,PEEK(175)-2:POKE104,PEEK(176)

(b) LOAD (SWAT program)

(c) POKE 103,1:POKE 104,8:DEL 0,0
```

A more complete description of these procedures can be found in the box on page 41 of last month's *SoftSide*.

ATARI®: Instead of SAVEing or CSAVEing the *SWAT* program to disk or tape, LIST it. The command would be LIST"D:SWAT" for disk, or LIST"C:" for cassette. You can then append the program to one in memory using either ENTER"D:SWAT" or ENTER"C:".

TRS-80®: If you have a disk, SAVE the *SWAT* program in ASCII format using the command SAVE"SWAT",A. It can then be appended to one in memory with the command MERGE"SWAT". Cassette users will need to execute the following series of commands:

```
(a) A=PEEK(16562)*256+254:A=A+65536*(A>3
2767):POKEA,PEEK(16548):POKEA+1,PEEK(165
49):A=PEEK(16633)+PEEK(16634)*256-2:B=IN
T(A/256):A=A-B*256:POKE16548,A:POKE16549
,B:CLEAR

(b) CLOAD (SWAT program)

(c) A=PEEK(16562)*256+254:A=A+65536*(A>3
2767):POKE16548,PEEK(A):POKE16549,PEEK(A
+1):CLEAR
```

Once you have appended *SWAT* to your typed-in program, just RUN it and it will produce a *SWAT* Table on your video screen or printer. On the Apple, type RUN 60000; on the ATARI®, type GOTO 32000; and on the TRS-80®, type RUN 65000. Once the *SWAT* Table has been generated, simply compare it to the one published with the program in the magazine. If it is identical you can pat yourself heartily on the back, delete the *SWAT* lines from memory, and get started using your program.

What to do if the *SWAT* Tables Don't Match

(1) First examine the listed line numbers in the first column. If they don't match, it probably means that you have inserted, omitted, or changed one or more line numbers (although a gross error in the length of a particular line could throw this column one entry off as well). An inserted or omitted line will affect all column one entries from that point on. Search the lines indicated by the earliest entry which is messed up, find any erroneous line number, and correct it. Then try running *SWAT* again to see if the entries match.

(2) If, after verifying all the line numbers, there are still discrepancies in columns two or three, more detailed trouble-shooting procedures are necessary. A bad entry in column three will virtually always be accompanied by a bad entry in column two, although the reverse may not be true.

(a) If the length entry in column three is bad, but the corresponding *SWAT* code in column two is ok, the most likely cause is a simple substitution of one character for another somewhere within the indicated lines. A variable named N0, for example, may have been typed as NO; a comma and a period may have been confused; a number such as 32767 may have been mistyped as 32757; or perhaps a word in a PRINT statement was misspelled. More complicated causes are also possible, with the same number of bytes added in one place as were omitted in another. Keep in mind that any BASIC keyword (PRINT, INPUT, FOR, etc., etc.) occupies only one byte of memory when stored in a program line. Thus, typing GOSUB instead of GOTO would not change the byte count in column three, but would change the *SWAT* code in column two.

(b) If the length entry in column three is bad, it's possible to get some clue about the nature of the error by noticing the size of the number compared to what it should be. A number which is too large usually indicates extraneous characters, while a smaller number usually indicates omitted characters. (Due to the way keywords are stored, this can be deceptive in some cases.) The only way to find the typing errors is simply to look for them, line by line and character by character. The advantage of using *SWAT* is that it narrows the field to no more than twelve lines or no more than about 500-700 bytes of code. The "resolution" of *SWAT* can easily be changed to a larger or smaller number of lines or bytes per grouping; and there will be occasions when we will publish a modified *SWAT* Table with greater resolution, for listings which invite more typographical errors than usual (such as this month's cover program).

Errors That Maybe Aren't

SWAT is very picky. That is its virtue, and its liability. If your typed-in program differs in ANY detail from that on our master disk, the tables won't match, even if those details are actually of no significance to the computer as it executes the program.

In particular, here are three types of picky details which you are likely to encounter.

(1) Differences in REM lines are treated by *SWAT* in exactly the same way as differences in other program lines. The majority of the programs we list contain few or no REMs; but if they DO have them, they must be entered exactly as listed. If you want to add REMs for your own benefit, wait until after you've *SWATted* the program to do so. Apple users: Remember that Applesoft always inserts one additional space following the keyword REM; so you should type in one LESS space than you see in a printed listing.

(2) Differences in DATA lines may or may not be significant. On the one hand, such lines are among the most trouble-prone, and are especially difficult to debug. *SWAT* can help here, perhaps more than anywhere else. On the other hand, extra spaces immediately before or after commas make no difference to the computer in reading data (unless they are enclosed in quotation marks) — but they DO make a difference to *SWAT*. If problems are indicated in lines which contain DATA statements, check not only the data items but their spacing. Again, Apple users must remember to type one less space following the keyword DATA than they see in the printed listing.

(3) Spacing, spelling, and punctuation within quotation marks are also important. In program instructions, for instance, the strings to be printed must be entered exactly as shown in the listing in order for *SWAT* to work reliably. Furthermore, in the case of the TRS-80®, all spaces between keywords, variables, etc., which are typed at the keyboard, are retained in memory and LISTed as entered. (The Apple and ATARI® delete unnecessary spaces when storing a program line in memory, and then format the line in a uniform way when LISTing it.) Therefore ALL spacing is critical on the TRS-80®, not just that which is included within quotation marks and in data elements.

Using *SWAT* To Check Itself

With two simple modifications you can use *SWAT* to verify itself once you've typed it into your computer. First, add one line to the end of the program:
Apple: 60200 REM
ATARI®: 32200 REM
TRS-80®: 65200 REM
Second, change the value of LN (in line 60000, 32000, or 65000) to equal the new line number you've just added. If you now run *SWAT* by itself (not appended to any other program) it should generate the table shown following the program listing for your computer.

Variables

A: Memory address.
A1: Beginning memory address for a group of program lines.
AA: Beginning memory address for a single program line.
B: Byte limit for memory scan.
C$: Two-letter *SWAT* Code.
C1, C2, C3, CQ: Used as constants; equal to 1, 2, 3, and 256.
D%: Used in converting S to a base-26 number.
D1%, D2%: Values of first and second digits of *SWAT* Code.
I, J: Loop counters.
L: Line number.
L1: Lowest line number of a group of lines.
L2: Highest line number of a group of lines.
LN: Beginning line number of the SWAT program.
N: Screen line counter for SWAT Table display.
NU: Line limit for memory scan.
PF: Printer flag; equals 1 if a printout is desired.
S: Sum of the contents of a range of memory locations.
X$: User input string.

APPLE™

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        APPLESOFT BASIC       $
$            'SWAT'            $
$     AUTHOR: JON VOSKUIL      $
$     (C) 1982     SOFTSIDE    $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialize variables. Clear the screen, and ask if a printout is desired.

```
60000 LN = 60000:NU = 12:B = 500
60010 A = 2049:C1 = 1:C2 = 2:C3 =
      3:CQ = 256:N = 0: HOME : PRINT
      "DO YOU WANT A PRINTOUT? ";:
       GET X$:PF = (X$ = "Y"): HOME
      : IF PF THEN  PR# 1: PRINT
```

Print the SWAT Table headings.

```
60020  PRINT "    LINES        SWA
      T CODE   LENGTH": PRINT "---
      ----------   ---------   ---
      ---"
```

Look in memory for the next program line number. If it is the first line of the appended SWAT program, then quit.

```
60030 L1 =  PEEK (A + C2) +  PEEK
      (A + C3) * CQ: IF L1 = LN THEN
       END
```

Adjust some variables.

```
60040 S = 0:A1 = A:L = L1
```

Begin a loop to search through NU program lines in memory.

```
60050  FOR I = 1 TO NU
```

Store the beginning address of the present line, and look in memory for the beginning address of the next line.

```
60060 AA = A:A =  PEEK (A) +  PEEK
      (A + C1) * CQ
```

Look in memory for the next program line number.

```
60070 L2 = L:L =  PEEK (A + C2) +
       PEEK (A + C3) * CQ
```

Add the contents of each byte of the current program line to S.

```
60080  FOR J = AA + C2 TO A - C1:
      S = S +  PEEK (J): NEXT
```

Check to see if the next line is the first line of the appended SWAT program, or if length has exceeded B bytes. If so, end the loop prematurely.

```
60090  IF L = LN OR A - A1 > B THEN
      I = NU
60100  NEXT I
```

Convert S to a base-26 number and express the last two digits as a two-letter code.

```
60110 D% = S / 676:S = S - D% * 6
      76:D1% = S / 26:D2% = S - D1
      % * 26:C$ =  CHR$ (D1% + 65)
      +  CHR$ (D2% + 65)
```

Print the three table entries for this group of lines. If the screen display mode is being used, then pause if the numbers are about to scroll off the screen.

```
60120  PRINT  TAB( 6 -  LEN ( STR$
      (L1)))L1" - "L2; TAB( 21)C$;
       TAB( 31)A - A1: IF  NOT PF THEN
      N = N + 1: IF N = 20 THEN N =
      1: PRINT : INPUT "PRESS RETU
      RN TO CONTINUE";X$: PRINT
```

Go back to search the next group of line numbers.

```
60130  GOTO 60030
```

APPLE SWAT TABLE FOR:
SWAT

LINES	SWAT CODE	LENGTH
60000 - 60110	SK	449
60120 - 60130	HG	110

ATARI®

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          ATARI BASIC         $
$            "SWAT"            $
$   AUTHOR: JON R. VOSKUIL     $
$     TRANSL: ALAN J. ZETT     $
$     (C) 1982     SOFTSIDE    $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialize variables. Find the starting address of the BASIC program, clear the screen, and ask if a printout is desired.

```
32000 CLR :LN=32000:NU=12:B=500
32010 CLOSE #1:CLOSE #2:A=PEEK(136)+PE
EK(137)*256:GRAPHICS 0:OPEN #2,4,0,"K:
":DIM Z$(12):Z$="            "
32020 C1=1:C2=2:C3=3:CQ=256:N=0:POSITI
ON 2,11:POKE 752,1:? "OUTPUT TO SCREEN
 OR PRINTER? (S/P)":GET #C2,X
```

Print the SWAT Table headings.

```
32030 TRAP 32020:OPEN #C1,8,0,CHR$(X):
POKE 752,C1:? #C1;"     LINES       SW
AT CODE   LENGTH"
32040 ? #C1;" -------------   --------
-    ------"
```

Look in memory for the next program line number. If it is the first line of the appended program, then quit.

```
32050 L1=PEEK(A)+PEEK(A+C1)*CQ:IF L1=L
N THEN CLOSE #C1:CLOSE #C2:POKE 752,0:
END
```

Adjust some variables.

```
32060 S=0:A1=A:L=L1
```

Begin a loop to search through NU program lines in memory.

```
32070 FOR I=C1 TO NU
```

Store the beginning address of the present line, and look in memory for the beginning address of the next line.

```
32080 AA=A:A=A+PEEK(A+2)
```

Look in memory for the next program line number.

```
32090 L2=L:L=PEEK(A)+PEEK(A+1)*CQ
```

Add the contents of each byte of the current program line to S.

```
32100 FOR J=AA TO A-C1:S=S+PEEK(J):NEX
T J
```

Check to see if the next line is the first line of the appended SWAT program, or if length has exceeded B bytes. If so, end the loop prematurely.

```
32110 IF L=LN OR A-A1>B THEN I=NU
32120 NEXT I
```

Convert S to a base-26 number.

```
32130 D=INT(S/676):S=S-D*676:D1=INT(S/
26):D2=S-D1*26
```

Print the three table entries for this group of lines, printing the last two digits of the base-26 number as a two-letter code.

```
32140 ? #1;Z$(1,6-LEN(STR$(L1)));L1;"
- ";L2;Z$(1,12-LEN(STR$(L2)));CHR$(D1+
65);CHR$(D2+65);
32150 ? #1;Z$(1,11-LEN(STR$(A-A1)));A-
A1:IF X<>83 THEN 32170
```

If the screen display mode is being used, then pause if the numbers are about to scroll off the screen.

```
32160 N=N+1:IF N=19 THEN N=1:? #1:? #1
;"RETURN TO CONTINUE":GET #2,D:? #1
```

Go back to search the next group of line numbers.

```
32170 GOTO 32050
```

ATARI® SWAT TABLE FOR:
SWAT

LINES	SWAT CODE	LENGTH
32000 - 32050	GR	518
32060 - 32160	UT	504
32170 - 32170	VC	13

TRS-80®

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$            TRS-80 BASIC         $
$               "SWAT"            $
$    AUTHOR: JON R. VOSKUIL       $
$      TRANSL: ALAN J. ZETT       $
$      (C) 1982      SOFTSIDE     $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialize variables. Find the starting address of the BASIC program and clear the screen. If a printout is desired, do the necessary POKEs.

```
65000 CLEAR99:LN=65000:NU=12:B=500
65010 A=PEEK(16548)+PEEK(16549)*256:N=0:P1=PEEK(16414):P2=PEEK(1
6415):CLS:PRINT@512,STRING$(63,143);:PRINT@528," Do you want a p
rintout? (Y/N) ";:GOSUB65130:CLS:IFX$="Y"THENPOKE16414,PEEK(1642
2):POKE16415,PEEK(16423)
```

Set up the PRINT USING string and print the SWAT Table headings.

```
65020 A$="##### - #####        %%        ####":PRINTTAB(17)"LINES
      SWAT CODE   LENGTH":PRINTTAB(13)"-------------   --------
-   ------"
```

Look in memory for the next program line number. If it is the first line of the appended SWAT program, then quit.

```
65030 L1=PEEK((A+2)+65536*(A+2>32767))+PEEK((A+3)+65536*(A+3>327
67))*256:A1=A:S=0:L=L1:IFL1=LNTHEN65120
```

Begin a loop to search through NU program lines in memory. Store the beginning address of the present line, and look in memory for the beginning address of the next line.

```
65040 FORI=1TONU:AA=A:A=PEEK((A)+65536*(A>32767))+PEEK((A+1)+655
36*(A+1>32767))*256
```

Look in memory for the next program line number.

```
65050 L2=L:L=PEEK((A+2)+65536*(A+2>32767))+PEEK((A+3)+65536*(A+3
>32767))*256
```

Add the contents of each byte of the current program line to S.

```
65060 FORJ=AA+2TOA-1:S=S+PEEK(J+65536*(J>32767)):NEXT
```

Check to see if the next line is the first line of the appended SWAT program, or if length has exceeded B bytes. If so, end the loop prematurely.

```
65070 IFL=LNORA-A1>BTHENI=NU
65080 NEXT
```

Convert S to a base-26 number and express the last two digits as a two-letter code.

```
65090 D=INT(S/676):S=S-D*676:D1=INT(S/26):D2=S-D1*26:C$=CHR$(D1+
65)+CHR$(D2+65)
```

Print the three table entries for this group of lines. If the screen display mode is being used, then pause if the numbers are about to scroll off the screen.

```
65100 PRINTTAB(13);USINGA$;L1;L2;C$;A-A1:IFX$<>"Y"THENN=N+1:IFN=
14THENN=0:PRINT:PRINT@960,STRING$(63,143);:PRINT@978," PRESS <EN
TER> TO CONTINUE ";:GOSUB65130:PRINT@896,CHR$(31);
```

Go back to search the next group of line numbers.

```
65110 GOTO65030
```

Restore print display to the screen and end.

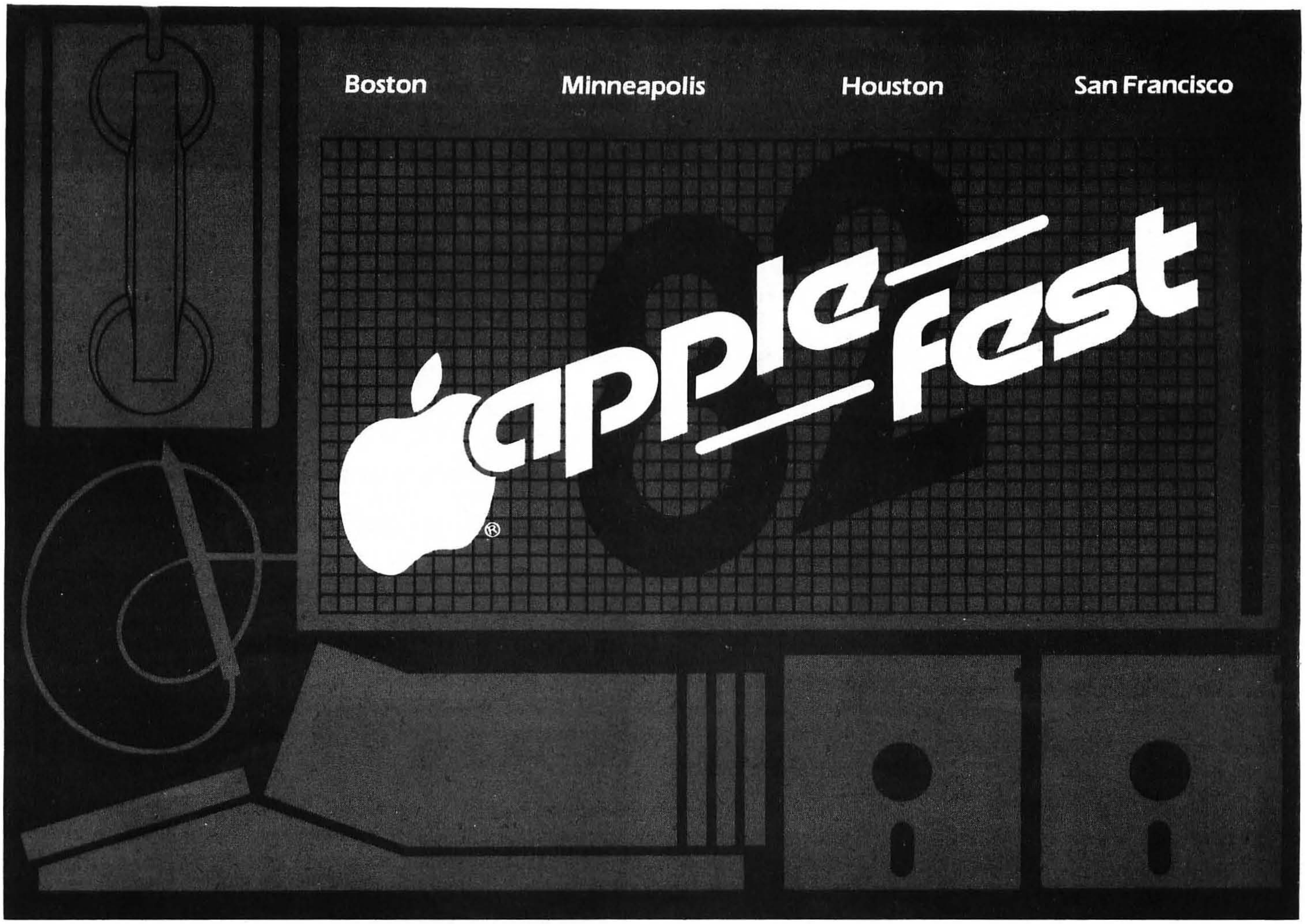
```
65120 PRINT:POKE16414,P1:POKE16415,P2:PRINT:END
```

Subroutine to get input characters.

```
65130 X$=INKEY$:IFX$=""THEN65130ELSERETURN
```

TRS-80® SWAT TABLE FOR: **SWAT**

LINES	SWAT CODE	LENGTH
65000 - 65050	JM	552
65060 - 65130	LK	334



The Most Spectacular Extravaganza Ever... For Apple Users

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

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

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

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

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

Ticket & Hotel Information

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

Exhibitor Information

For specific exhibitor information on one or all of the Applefest '82 shows call Northeast Expositions at the telephone number above.

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

Applefest/Minneapolis
Thurs-Sun Sept 16-19, 1982
Minnesota Auditorium and Convention Hall
Show Hours: 11 AM to 6 PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days, $15 for 4 days

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

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

Applefest is produced by Northeast Expositions Inc. and is sanctioned by Apple Computer Inc. and The Boston Computer Society.

*Apple and Applefest are registered trade and service marks of Apple Computer Inc.

NORMAL LOADING PROCEDURES FOR SOFTSIDE DISKS AND CASSETTES

Follow these procedures unless otherwise instructed by the documentation in the magazine. Back issues may differ in some details.

APPLE

Disks are in 13-sector format, created under DOS 3.2.1. If your system is set up for 16-sector disks (DOS 3.3), first boot your BASICS disk or BRUN BOOT13 from the System Master Diskette, and then insert the ***SoftSide*** disk. A cover/menu program will run automatically.

Tapes LOAD in the normal way. Advance the tape to the beginning of the lead-in tone; stop the tape; insert the plug into the EAR jack; type LOAD; start the tape; and press RETURN. Side two of the tape is a duplicate of side one, unless one or more Integer BASIC programs are included, in which case side two contains the Integer programs.

ATARI®

Disks do not contain DOS.SYS files, and are therefore not bootable by themselves. First boot a disk which contains any version of DOS, then insert the ***SoftSide*** disk and RUN "D:COVER".

Tapes CLOAD in the normal way. If you have difficulty, try this procedure:
(1) Type POKE 54018,54 and press RETURN.
(2) Turn up the volume on your TV.
(3) Type CLOAD and press RETURN once.
(4) Press the PLAY button and listen.
(5) When you hear a steady lead-in tone, press RETURN again.
Side two of the tape is a duplicate of side one.

TRS-80®

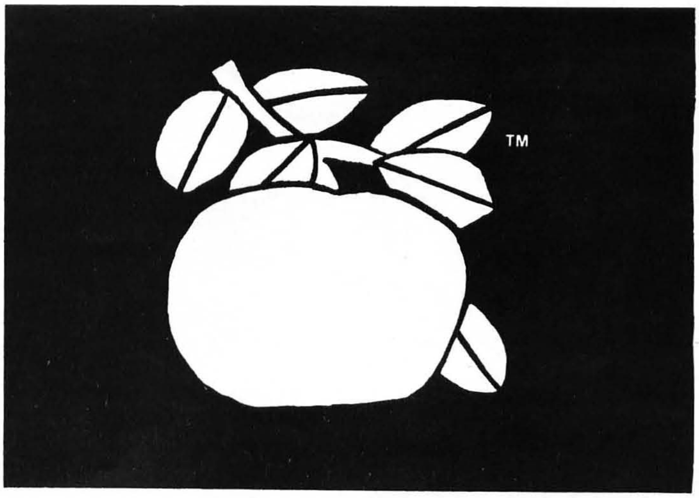
Disks are available in Model I or Model III format. They contain the DOS PLUS operating system, and a cover program which automatically runs upon booting. Back issues prior to May, 1982, are available only in Model I format, and may be converted using the TRSDOS CONVERT utility on a two-drive Model III. Older back issues (with Model I TRSDOS) require you to enter BASIC and then type RUN "COVER".

Tapes CLOAD in the normal way on Model I's, and at low speed (500 baud) on Model III's. The first program is a cover/menu program. Side two of the tape is a duplicate of side one.

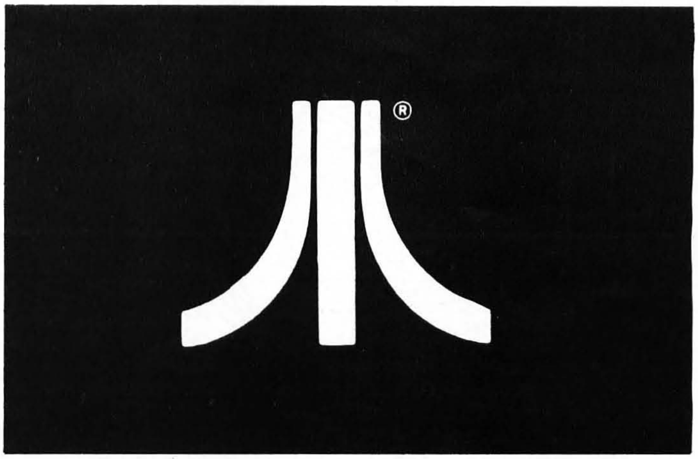
GENERAL NOTES ABOUT MAGNETIC MEDIA

Our disks and tapes are duplicated by reliable, professional duplication services; bad copies are now very rare. However, the trip through the mail does occasionally wreak havoc with sensitive magnetic media. If, after a reasonable number of tries and a careful check and cleaning of your equipment, you are not able to load a program from a tape or disk, please return it to us with an exact description of the problem. If we cannot duplicate the problem on our systems, we will advise you of that when we send the replacement.

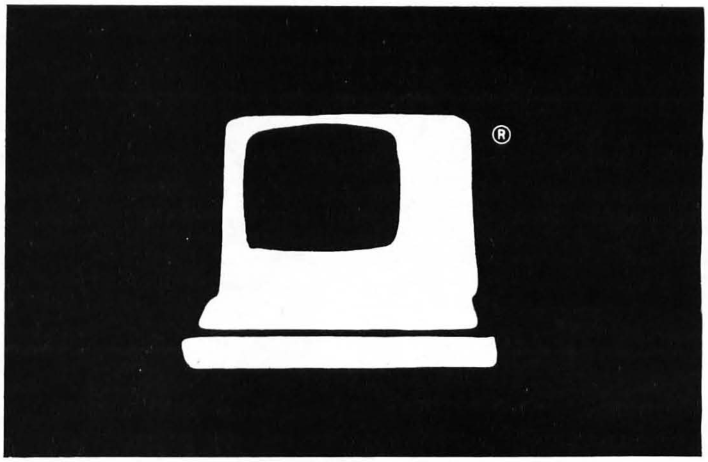
We use no copy-protection schemes on our media. We urge you to make a backup copy of every disk or tape as soon as you receive it (and at the same time resist the urge to give copies to friends). Our replacement policy does not extend beyond 30 days. ꩜



APPLE™/SIDE
page______54

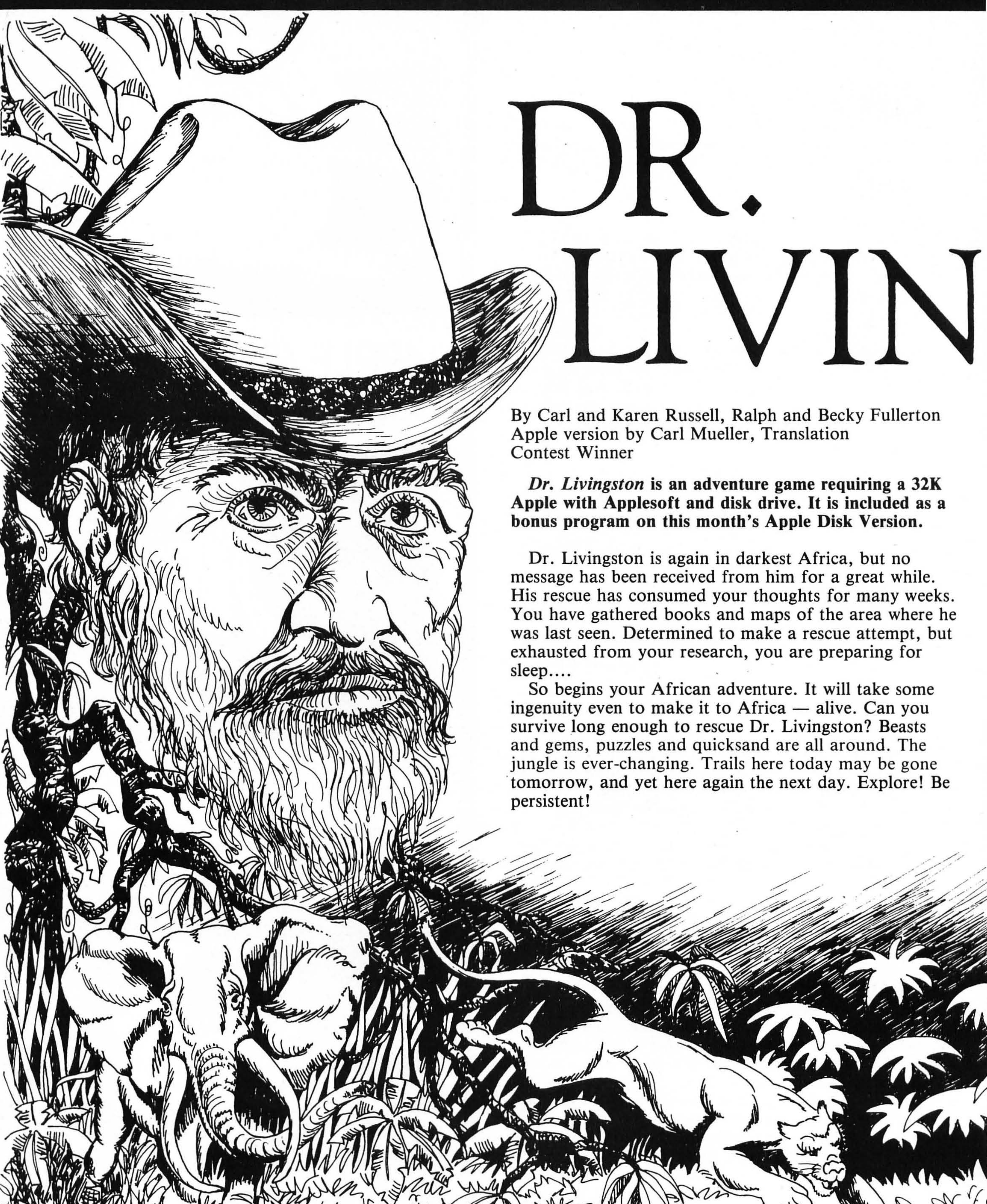


ATARI®/SIDE
page______73



TRS-80®/SIDE
page______83

DR. LIVIN



By Carl and Karen Russell, Ralph and Becky Fullerton
Apple version by Carl Mueller, Translation
Contest Winner

***Dr. Livingston* is an adventure game requiring a 32K Apple with Applesoft and disk drive. It is included as a bonus program on this month's Apple Disk Version.**

Dr. Livingston is again in darkest Africa, but no message has been received from him for a great while. His rescue has consumed your thoughts for many weeks. You have gathered books and maps of the area where he was last seen. Determined to make a rescue attempt, but exhausted from your research, you are preparing for sleep....

So begins your African adventure. It will take some ingenuity even to make it to Africa — alive. Can you survive long enough to rescue Dr. Livingston? Beasts and gems, puzzles and quicksand are all around. The jungle is ever-changing. Trails here today may be gone tomorrow, and yet here again the next day. Explore! Be persistent!

APPLE DV

GSTON

Your African adventure is about to begin.

To explore the terrain and locate the good doctor, you need to enter two-word commands such as "GET BOOK" and "GO EAST". You may also move by entering just the initial letter of the direction: E, W, N, S, U, or D. The computer's vocabulary is limited, so if a word is not accepted, try a similar word or a new approach. Here are five helpful one-word commands:

HINT: Will give a clue in some circumstances.

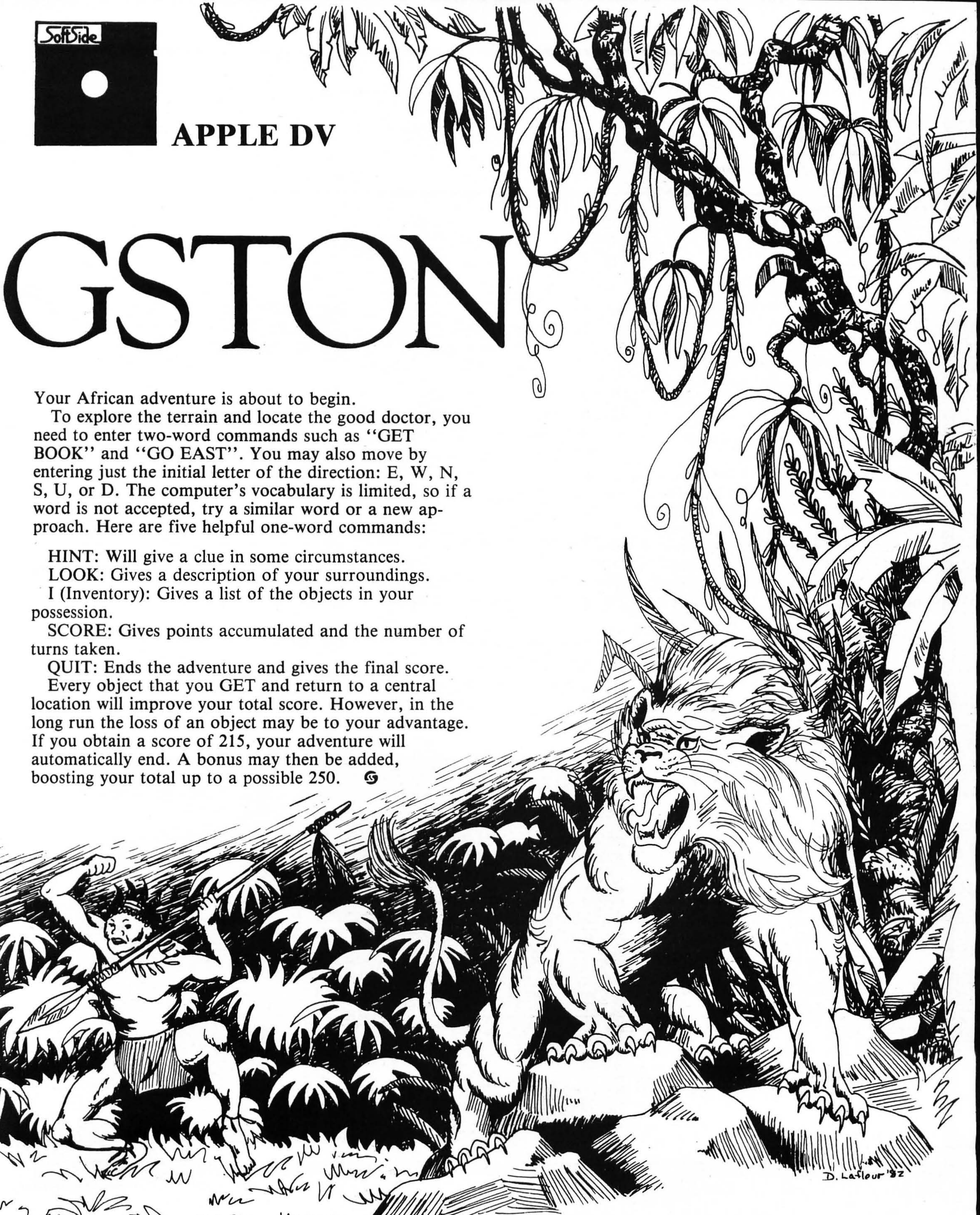
LOOK: Gives a description of your surroundings.

I (Inventory): Gives a list of the objects in your possession.

SCORE: Gives points accumulated and the number of turns taken.

QUIT: Ends the adventure and gives the final score.

Every object that you GET and return to a central location will improve your total score. However, in the long run the loss of an object may be to your advantage. If you obtain a score of 215, your adventure will automatically end. A bonus may then be added, boosting your total up to a possible 250.



K-Byter

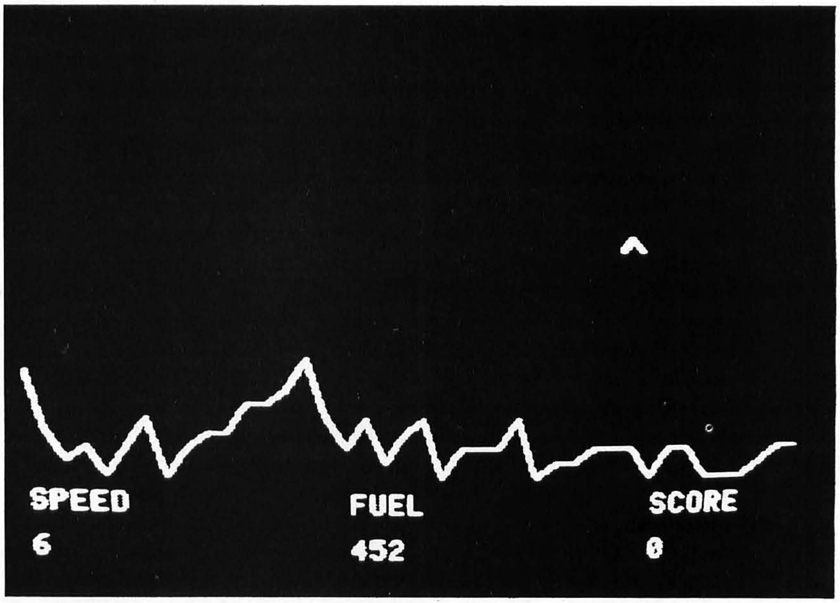
Space Float

An Applesoft K-Byter by Peter Wu, Montgomery, WV

Nearly everyone has played some form of lunar lander game. This variation has a unique "feel" to it which the ***SoftSide*** staff found very appealing — especially considering its compact code.

The object of the game is to score as many points as possible by landing your spacecraft on flat terrain. The number of points you receive depends on the speed of your craft at landing. If you crash, the computer will beep and subtract 75 of your 500 units of fuel. Use Paddle 0 to rotate your ship, and the paddle button to apply thrust.

```
10  HOME : VTAB 21: PRINT "SPEED"
      ,"FUEL","SCORE": POKE 34,21
20  HGR : HCOLOR= 3: SCALE= 6: ROT=
      8:F = 500: DIM X(42): GOSUB
      310
30  FOR P = 1 TO 1000: NEXT : IF
      F <  = 0 THEN  END
40  HGR :Y = 24: HCOLOR= 3: HPLOT
      0,Y * 5
50  VTAB 23: PRINT  INT (YM * 7)"
       ",F" ",S
60 X(0) = 120:X(1) = X(0)
70  FOR X = 2 TO 41
80 T =  INT ( RND (1) * 6) - 2: IF
      T >  = 1 THEN T = T + 1
90 Y = Y + T
100  IF Y > 31 THEN Y = 29
110  IF Y < 21 THEN Y = 23
120  HPLOT  TO (X - 1) * 7 - 1,Y *
      5
130 X(X) = Y * 5
140  NEXT
150 X(42) = Y * 5
160 X = 10:Y = 10:YM = 0:XM = 1:D
      =  - 16286
170 R =  INT ( PDL (1) / 15)
180  IF  PEEK (D) > 127 AND F > 0
       THEN YM = YM - .22 +  ABS (
      R - 8) / 50:XM = XM + (R - 8
      ) / 128:F = F - 1
190  VTAB 23: PRINT  INT (YM * 7)
      " ",F" "
200 YM = YM + .05: IF YM > 10 THEN
      YM = 10
210 Y = Y + YM:X = X + XM
220  IF Y < 1 THEN YM =  - YM:Y =
      1
230  IF X < 3 THEN X = 277
240  IF X > 277 THEN X = 3
250  HCOLOR= 3: ROT= R: DRAW 1 AT
      X,Y
260  HCOLOR= 4: ROT= RO: DRAW 1 AT
      OX,OY
270  HCOLOR= 3: ROT= R: DRAW 1 AT
      X,Y
280 T =  INT (X / 7 + .5) + 1: IF
      Y + 5 > X(T) THEN  IF YM < 1
       THEN  IF  ABS (R - 8) < 2 THEN
       IF X(T) = X(T + 1) AND X(T)
       = X(T - 1) THEN S = S +  INT
      (( - YM + 1) * 100): GOTO 30

290  IF Y + 3 > X(T) THEN F = F -
      75: PRINT  CHR$ (7);: GOTO 3
      0
300 OX = X:OY = Y:RO = R: GOTO 17
      0
310  FOR X = 768 TO 774: READ Y: POKE
      X,Y: NEXT : POKE 232,0: POKE
      233,3: RETURN
320  DATA    01,00,04,00,57,38,0
```



SPEED FUEL SCORE
6 452 0

CODE BREAKER

by Steve Faiella

***Codebreaker* is a color graphics game of logic for a 16K Apple with Applesoft.**

If you are the kind of person who enjoys games of logic and deduction, then *Codebreaker* is for you. The description of the game is relatively simple. The computer generates a random "code", a sequence of four colored pegs, which the player must try to guess. The colors available to the Apple for the code are red, yellow, orange, green, tan, and blue. There may be duplication of colors in a code, such as red/yellow/red/blue or orange/yellow/yellow/yellow. Using the six colors, with duplication allowed, there are almost 1300 possible combinations.

Take heart, the Apple gives you clues along the way. Every time you input a guess, the Apple awards you "hint pegs" in the blue hint board to the left. For every color you guess which is the same COLOR as a peg in the computer's code, you get one white hint peg. For every color you guess which is the same COLOR AND in the same POSITION as a peg in the computer's code, you get one black hint peg. By observing your past guesses and the hint pegs awarded, you can arrive at the hidden code.

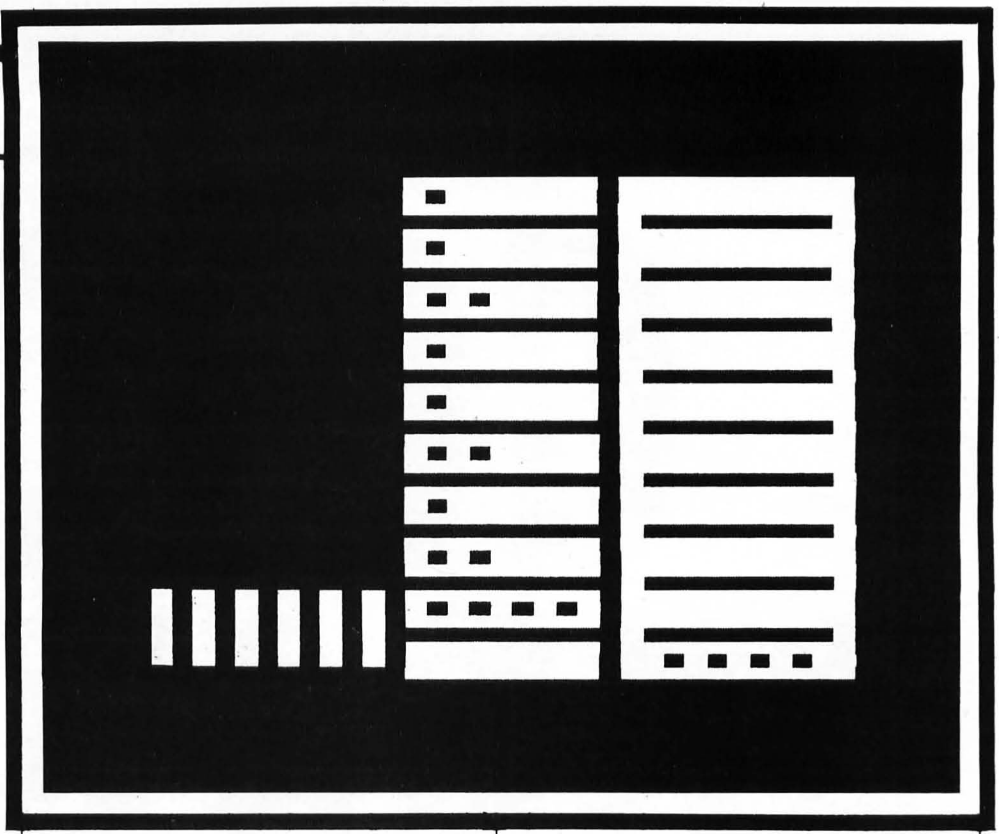
Using the Program

After the title is displayed, you are given the option of seeing the instructions. Following this the guess board and hint board are displayed. Before each round the Apple beeps once to signal that it is ready. You input your guess for the code, one peg at a time, by entering the first letter of the color names, which are listed at the left of the screen. If, during your input, you want to change your guess for that round, simply enter "E" instead of a color initial, and you will be prompted for peg #1 again. After all four peg colors have been entered, they are displayed and you are given one more chance to change your mind. The colors are then filled in on the graphics display, along with the appropriate hint pegs. If your guess is correct, you are given a score based on the number of rounds it took you to break the code, and a "luck factor" based on the number of black and white pegs you were awarded. If you want to end the game at any time, just press "Q" during any peg input.

Variables

C(5): Color code generated by the computer, changed during execution.
CA(7): Array of color numbers.
CC$(13): Array of color names.
G(4): Array of player's guesses converted to numbers.
G1(4): Array of player's guesses used for comparison.
H(5): Array of colors (a copy of the C array which is not altered).
I: Loop variable.
IS$, PL$: Input strings.
POK$: Used to hold data to poke in for the sound routine.
PS: Used to choose random colors.
PV(4): Array of black and white pegs.
SC: Score.
W$, WN, X, X1, X2, X2$, X3, X4, X5: Miscellaneous work variables.
Y: Main game variable.

APPLE™



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$        'CODEBREAKER'         $
$   AUTHOR: STEVE FAIELLA      $
$    (C) 1982    SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Display color patch test.

```
5  GOSUB 2000
```

Print title and instructions if desired.

```
10  GOSUB 1020: HOME
```

Poke in the Machine Language ampersand sound routine.

```
20  GOSUB 910
```

Initialize variables.

```
30  DIM CC$(13),H(5),C(5),CA(7)
40 CC$(1) = "RED":CC$(2) = "BLU":
     CC$(4) = "GRN":CC$(8) = "TAN
     ":CC$(9) = "ORG":CC$(13) = "
     YEL"
50  GOSUB 730
60  FOR X1 = 1 TO 6: READ CA(X1):
      NEXT X1
65  DATA 1,9,13,4,2,8
```

Set up the board graphics.

```
70  GOSUB 800
80  HOME : PRINT "T O G B Y R"
90  PRINT "A R R L E E"
100  PRINT "N G N U L D";
```

Generate a random color code.

```
120  HTAB 15: INVERSE : PRINT "GE
      NERATING CODE...";: NORMAL
130  FOR X1 = 1 TO 4
140 PS = ( INT ( RND (1) * 6) + 1
     )
150 C(X1) = CA(PS)
160 H(X1) = CA(PS)
170  NEXT X1
180  FOR X = 250 TO 20 STEP  - 2:
      & TX,7: NEXT X
```

Main game loop.

```
190  FOR Y = 1 TO 10
200  VTAB 23: HTAB 15: PRINT "ROU
     ND ";Y;"                  ";
210  FOR X = 1 TO 2000: NEXT X
220  & T80,100
230 X2 = 1
240  VTAB 24: HTAB 15: PRINT "E=C
     ORRECT ERROR";
250  VTAB 23: HTAB 15: PRINT "GUE
     SS FOR PEG ";X2;" ";: GET X2
     $
270  IF X2$ = "E" THEN  GOTO 230
280  IF X2$ = "T" THEN G(X2) = 8:
      GOTO 350
290  IF X2$ = "O" THEN G(X2) = 9:
      GOTO 350
300  IF X2$ = "G" THEN G(X2) = 4:
      GOTO 350
310  IF X2$ = "B" THEN G(X2) = 2:
      GOTO 350
320  IF X2$ = "Y" THEN G(X2) = 13
     : GOTO 350
330  IF X2$ = "R" THEN G(X2) = 1:
      GOTO 350
335  IF X2$ = "Q" THEN  TEXT : HOME
     : END
340  FOR X = 1 TO 50: & T200,5: NEXT
      X: GOTO 250
350  PRINT CC$(G(X2));: & T200,15
     : FOR I = 1 TO 300: NEXT I: HTAB
     31: PRINT "   ";
360 X2 = X2 + 1: IF X2 < 5 THEN  GOTO
     250
370  VTAB 23: HTAB 15: FOR I = 1 TO
     4: PRINT CC$(G(I));" ";: NEXT
     I
380  VTAB 23: HTAB 30: PRINT "-OK
     (Y/N)";: GET W$
390  IF W$ = "N" THEN  VTAB 23: HTAB
     30: PRINT "          ";: GOTO
     230
400  IF W$ < > "N" AND W$ < > "
     Y" THEN 370
410  VTAB 23: HTAB 15: PRINT "
                              ";
420  VTAB 24: HTAB 15: PRINT "
                    ";
430  GOSUB 460: GOSUB 550: GOSUB
     730
440  NEXT Y
```

Plot the guessed colors on the screen.

```
450 WN = 2: GOSUB 750
460 I = 1
480 X5 = ((Y - 1) * 4) + 1
490  FOR X3 = 24 TO 30 STEP 2
500  COLOR= G(I)
510  PLOT X3,X5: & T( INT ( RND (
     1) * 50) + 50),10
520 I = I + 1
```

```
530  NEXT X3
540  RETURN
```

Award black and white hint pegs.

```
550 X1 = 1
560  FOR X = 1 TO 4
570  IF G(X) = C(X) THEN PV(X1) =
      0:C(X) = 255:G(X) = 254:X1 =
      X1 + 1:SC = SC + 10
580  NEXT X
590  FOR X = 1 TO 4
600  FOR X2 = 1 TO 4
610  IF G(X) = C(X2) THEN PV(X1) =
      15:C(X2) = 255:G(X) = 254:X1
       = X1 + 1:SC = SC + 1: GOTO
      630
620  NEXT X2
630  NEXT X
640 I = 1
650 X3 = ((Y - 1) * 4) + 1
660  FOR X4 = 13 TO 19 STEP 2
670  COLOR= PV(I): PLOT X4,X3
680 I = I + 1
690  IF PV(I) <  > 7 THEN  & T50,
      10: FOR X5 = 1 TO 200: NEXT
      X5
700  NEXT X4
710  IF PV(1) = 0 AND PV(2) = 0 AND
      PV(3) = 0 AND PV(4) = 0 THEN
      WN = 1: GOSUB 750
720  RETURN
```

Restore peg arrays.

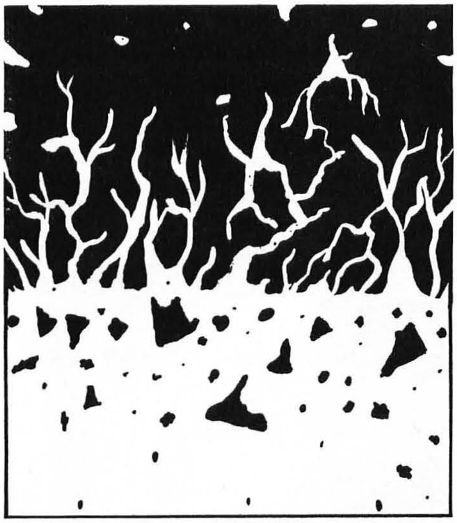
```
730  FOR X3 = 1 TO 4:PV(X3) = 7:C
      (X3) = H(X3):G(X3) = G1(X3):
       NEXT X3
740  RETURN
```

Win/lose and end-game routine.

```
750  IF WN = 2 THEN  HOME : PRINT
      "SORRY, YOU LOSE!  THE COLOR
      S WERE:": PRINT CC$(H(1));"
      ";CC$(H(2));" ";CC$(H(3));"
      ";CC$(H(4)): GOSUB 980: GOTO
      780
760 SC = SC + ((10 - Y) * 100): HOME
      : PRINT "YOU WIN!!! YOUR SCO
      RE WAS ";SC
770  GOSUB 980
780  INPUT "PLAY AGAIN (Y/N)?";PL
      $: IF PL$ = "Y" THEN  RUN 30
790  TEXT : HOME : END
```

Subroutine to draw the board.

```
800  GR : COLOR= 15
820  FOR X = 23 TO 31: VLIN 0,38 AT
      X: NEXT
830  COLOR= 10: VLIN 0,38 AT 22: VLIN
      0,38 AT 32
840  COLOR= 0: FOR X = 3 TO 36 STEP
      4: HLIN 23,31 AT X: NEXT X
850  COLOR= 7: FOR X = 12 TO 20: VLIN
      0,38 AT X: NEXT X
860  COLOR= 0: FOR X = 3 TO 36 STEP
      4: HLIN 12,20 AT X: NEXT X
870  COLOR= 0: FOR X = 1 TO 37 STEP
      4: PLOT 24,X: PLOT 26,X: PLOT
      28,X: PLOT 30,X: NEXT X
```



```
880  COLOR= 8: VLIN 32,37 AT 0: COLOR=
      9: VLIN 32,37 AT 2: COLOR= 4
      : VLIN 32,37 AT 4: COLOR= 2:
      VLIN 32,37 AT 6: COLOR= 13:
      VLIN 32,37 AT 8: COLOR= 1: VLIN
      32,37 AT 10
890  RETURN
```

Subroutine to poke in the sound routine.

```
910 POK$ = "201,084,208,015,032,1
      77,000,032,248,230,138,072,0
      32,183,000,201,044,240,003,0
      76,201,222,032,177,000,032,2
      48,230,104,134,003,134,001,1
      33,000"
920  FOR I = 1 TO 35: POKE I + 76
      7, VAL ( MID$ (POK$,I * 4 -
      3,I * 4 - 1)): NEXT I
930 POK$ = "170,160,001,132,002,1
      73,048,192,136,208,004,198,0
      01,240,007,202,208,246,166,0
      00,208,239,165,003,133,001,1
      98,002,208,241,096"
940  FOR I = 1 TO 33: POKE I + 80
      2, VAL ( MID$ (POK$,I * 4 -
      3,I * 4 - 1)): NEXT I
950  POKE 1013,76: POKE 1014,0: POKE
      1015,3
960 POK$ = "":I = 0
970  RETURN
```

Play the winning or losing song.

```
980  IF WN = 1 THEN 1000
990  FOR X = 1 TO 100: & T100,3: NEXT
      : GOTO 1010
1000  & T89,150: & T89,150: & T10
      1,75: & T108,75: & T108,150:
       & T112,75: & T108,75: & T10
      8,255
1010  RETURN
```

Subroutine to print the instructions.

```
1020  TEXT : HOME : VTAB 12: HTAB
      9: PRINT "C O D E - B R E A
      K E R"
1030  VTAB 23: HTAB 9: PRINT "NEE
      D INSTRUCTIONS (Y/N)";: GET
      IS$: IF IS$ = "N" THEN  RETURN

1040  IF IS$ <  > "Y" THEN 1030
1060  HOME : PRINT "IN CODE BREAK
      ER, THE APPLE SETS UP A    P
      ATTERN USING FOUR COLORED 'P
      EGS'. IT   IS UP TO YOU TO D
      ETERMINE THE COLORS OF THE P
      EGS AND THE SEQUENCE IN WHIC
      H THEY WERE CHOSEN."
1070  PRINT : PRINT "YOU ARE GIVE
      N HINTS ALONG THE WAY VIA
      THE HINT BOARD AT THE LEFT.
      FOR EVERY   COLOR THAT YOU G
      UESS CORRECTLY, THE     APPL
      E PLACES A WHITE PEG IN THE
      HINT"
```

```
1080  PRINT "BOARD.  FOR EVERY CO
     LOR YOU GUESS WHICH IS ALSO
     IN THE CORRECT SEQUENCE, YOU
         WILL GET A BLACK PEG.": PRINT
     : PRINT "FOR EXAMPLE, IF THE
      APPLE CHOOSES:"
1090  INVERSE : HTAB 3: PRINT "
     RED     YELLOW     BLUE     GRE
     EN  ": NORMAL
1100  PRINT "AND YOUR GUESS IS:"
1110  INVERSE : HTAB 3: PRINT "
     RED     TAN      ORANGE    BLU
     E   ": NORMAL
1120  PRINT "YOU RECEIVE 1 BLACK
     PEG (FOR RED) AND 1 WHITE PE
     G (FOR BLUE)."
1130  VTAB 23: INPUT "HIT RETURN
     TO CONTINUE";IS$
1140  HOME : INVERSE : PRINT "THE
      ORDER OF THE WHITE AND BLAC
     K PEGS   IS NOT IMPORTANT.
                               ": NORMAL
1150  PRINT "THE APPLE ALWAYS GIV
     ES OUT BLACK PEGS   FIRST; S
     O, IF YOU GET ONE BLACK AND
     TWO WHITE PEGS, IT MIGHT NOT
      BE THE FIRST   COLOR IN YOU
     R GUESS WHICH IS IN THE
     PROPER SEQUENCE."
1155  PRINT : PRINT "ANOTHER THIN
     G TO BE AWARE OF IS THAT
     THE CODE MAY CONTAIN DUPLICA
     TE COLORS.  A VALID CODE COU
     LD BE:": PRINT : PRINT "
     TAN     YELLOW     TAN     RED.
     "
1160  PRINT : PRINT "YOU ARE GIVE
     N TEN ROUNDS TO BREAK THE
     APPLE'S CODE. IF YOU GUESS I
     T IN TEN    ROUNDS OR LESS,
     YOU WIN, AND ARE SCORED BASE
     D ON THE NUMBER OF BLACK AND
      WHITE  PEGS YOU WERE AWARDE
     D, AND HOW MANY"
1170  PRINT "ROUNDS IT TOOK YOU T
     O BREAK THE CODE."
1180  PRINT : INPUT "HIT RETURN T
     O CONTINUE";IS$: RETURN
```

Subroutine to display color patch test.

```
2000  GR : COLOR= 15: FOR X = 1 TO
     35: HLIN 0,39 AT X: NEXT X
2010  COLOR= 1: FOR X = 9 TO 13: HLIN
     3,8 AT X: NEXT X
2020  COLOR= 9: FOR X = 9 TO 13: HLIN
     18,23 AT X: NEXT X
2030  COLOR= 13: FOR X = 9 TO 13:
     HLIN 33,38 AT X: NEXT X
2040  COLOR= 4: FOR X = 20 TO 24:
     HLIN 3,8 AT X: NEXT X
2050  COLOR= 2: FOR X = 20 TO 24:
     HLIN 18,23 AT X: NEXT X
2060  COLOR= 8: FOR X = 20 TO 24:
     HLIN 33,38 AT X: NEXT X
2080  HOME : PRINT "COLOR TEST PA
     TCH - ADJUST THE COLOR AND B
     RIGHTNESS ON YOUR SET SO YOU
      HAVE:"
2090  PRINT  TAB( 10)"RED  ORANGE
      YELLOW"
2100  PRINT  TAB( 9)"GREEN  BLUE
      TAN";
2200  HTAB 32: PRINT "ANY KEY";: GET
     W$: RETURN
```

APPLE SWAT TABLE FOR:
CODEBREAKER

LINES	SWAT CODE	LENGTH
5 - 100	CV	263
120 - 230	JK	230
240 - 350	NR	347
360 - 480	FC	296
490 - 600	YC	191
610 - 720	NO	249
730 - 850	HS	403
860 - 940	VO	533
950 - 1070	PM	643
1080 - 1150	TO	658
1155 - 2020	IE	519
2030 - 2200	LS	280

COMMANDing BASIC

String-Defined Functions In Applesoft

by Michael Prescott

The normal way to define a function in an Applesoft program is to use the DEF FN statement in a program line. If you want to plot a graph on the Hi-Res screen, for example, you might include a line such as the following in your program:

```
100 DEF FN Y(X) = X + 10 *
SIN(X) + 3.14159
```

Then, instead of using the expression "X + 10 * SIN (X) + 3.14159" in calculating values, you can simply use the expression "FN Y(X)".

Suppose, however, that you want a program which will be able to make use of any function that the user might want to specify. This would usually involve the somewhat awkward procedure of stopping execution of the program, having the user type in a new program line in the proper format, and then restarting the program. Although this does work, it's cumbersome and not very elegant. The routine described here can be incorporated into your own Applesoft program, and will allow your program to accept function-definition statements in the form of input strings, without having to stop execution and change program lines. Since the function definitions are stored in the same manner as normal strings, they can be preserved (in disk text files, for example) to form a library of functions accessible to any number of programs.

Two forms of the routine are listed at the end of this introduction. The first consists of a commented Assembly Language listing of the code for those who wish to examine its operation more closely; the second consists of an Applesoft program which POKEs the Machine Language routine into memory and demonstrates its use.

How the Routine Works

The demonstration program will clear the screen, display a title, and then initialize the Machine Language routines in memory, starting at decimal address 768. Although this is a convenient location in many respects, you should be aware that the normal DOS entry point at 3D0 will be overwritten. (Consult page 144 of the DOS manual for alternate entry points.) When using the routine in your own programs, you may prefer to use a higher memory location, protected using appropriate LOMEM or HIMEM statements.

Once the Machine Language routines have been initialized, they are ready to be used. These routines are intended to be used under deferred execution; that is, during the running of a program. They will work under immediate execution (i.e., typing commands on the keyboard), but doing this will cause strange side effects to occur.

To start, we need a string, containing some combination of numeric and logical expressions, which defines the function desired. The rules of syntax are exactly the same as for Applesoft's DEF FN statement. You are not limited to using one variable, however. In the demo program, lines 120-130 prompt you for an expression which uses two variables, X and Y. In response you might enter the expression "X*2 + Y", for example. This string, although legible to you as a mathematical/logical expression, is not yet legible to the Apple as anything other than a normal string. This is where the String-Defined Function routine comes in. Once your expression is contained in memory, the statement "& DEF var$" in your program (see line 140) will replace your ASCII string with a string of tokens recognized by the Apple.

Once this has been done, the string will be in a form which is quite unintelligible to the user, since it has literally become a line of program text. However, it is possible to list the string. To do this type "&LIST var$" into your program, as in line 180 of the demo program. This will cause your string to be printed on the screen, in much the same way that a program line is listed.

The side effect of using "&DEF" and "&LIST" during immediate execution is that you will receive a SYNTAX ERROR message, although the routines are functioning properly. This is because these routines use an area of memory which is also used by the Apple to get input text. There is no way to avoid this.

Once we have a string that the Apple can recognize as a function, we can use it just as we would a FN statement, except that it must be in the form of a "USR (var$)" statement. The USR statement can be used in either immediate or deferred execution without any side effects. After prompting you for the values of X and Y in line 150, line 160 of the demo program shows the use of the USR statement to invoke the defined function, assigning the result to the variable Z.

The only difference between the Apple's FN and the String-Defined Function is that no argument (i.e., the X of FN(X)) will be passed on to the function. Rather, the String-Defined Function will *do its calculations* with the values already as-

```
                *                              *
                * STRING-DEFINED FUNCTION PROG *
                *                              *
                *
                CH        EQU   $24
                REGL      EQU   $85
                REGH      EQU   $86
                LOWTRL    EQU   $9B
                LOWTRH    EQU   $9C
                CHRGET    EQU   $B1
                TXPTRL    EQU   $B8
                TXPTRH    EQU   $B9
                INBUFF    EQU   $200
                ERROR     EQU   $D412
                TKNIZE    EQU   $D56C
                LISTPR    EQU   $D6FC
                FRMNUM    EQU   $DD67
                CHKSTR    EQU   $DD6C
                SYNERR    EQU   $DEC9
                PTRGET    EQU   $DFE3
                GETSPA    EQU   $E452
                MOVSTR    EQU   $E5E2
                *
                          ORG   $300
                *
                * &DEF ROUTINE
                *
0300: C9 B8               CMP   #$B8      IS IT 'DEF'?
0302: D0 6B               BNE   LIST      NOPE, TRY 'LIST'.
0304: 20 B1 00            JSR   CHRGET    YES, GOBBLE FIRST CHAR.
0307: 20 E3 DF            JSR   PTRGET    FIND VAR. AND SAVE ITS ADDR.
030A: 20 6C DD            JSR   CHKSTR    MAKE SURE IT'S A STRING.
030D: A5 B8               LDA   TXPTRL    FETCH TXTPTR
030F: 48                  PHA             AND SAVE IT
0310: A5 B9               LDA   TXPTRH    FOR LATER.
0312: 48                  PHA
0313: A0 02               LDY   #$02
0315: B1 83               LDA   ($83),Y   GET STRING ADDRHI
0317: 85 86               STA   $86       AND SAVE IT
0319: 88                  DEY
031A: B1 83               LDA   ($83),Y   LIKEWISE STRING ADDRLO...
031C: 85 85               STA   REGL
031E: 88                  DEY
031F: B1 83               LDA   ($83),Y   ...AND THE STRING LENGTH
0321: A8                  TAY
                *
                * COPY THE STRING TO THE INPUT BUFFER
                * AND SET LAST CHARACTER TO 00
                *
0322: A9 00               LDA   #$00
0324: 99 00 02            STA   INBUFF,Y
0327: 88        LOOP1     DEY
0328: C0 FF               CPY   #$FF      FINISHED?
032A: F0 09               BEQ   TOKEN     YES, GO ON
032C: B1 85               LDA   (REGL),Y  NO, GET NEXT CHAR.
```

signed to the variables involved. String-Defined Functions may be nested indefinitely; but, when doing this, be certain that none of the functions call themselves. If a function does call itself an OUT OF MEMORY error will result.

The uses of this program range from interactive graphing applications, to business programs where ease of changing a few formulas is important. The string functions are identical to normal Applesoft strings and may be saved on disk or tape; once defined, they need not be redefined. In this way a database of useful functions can be stored on disk and read in on command, or stored in string arrays.

The program could also be used within another program which needs to change its own function parameters while running. This type of application could occur in the field of industrial control. In this case the normal Apple string functions could be used to redefine the string and thus the function. The program might also be used in artificial intelligence applications.

The String-Defined Functions routine depends heavily on the built-in routines within the Apple's ROM. First the ampersand (&) effects a call to memory location $3F5. At this address is the instruction to jump to $300 (as set up by the default address value). The A register is then loaded with the ASCII value of the character following "&". At $300 there is a command to check whether the A register contains the token for "DEF"; if not, then it jumps to $36F, where it checks for the "LIST" tokén. By this method the proper subroutine is accessed.

The USR function, as explained on page 45 of the Applesoft manual, takes the argument and puts its value into the Floating-point ACcumulator, FAC for short. Since we are dealing with strings, it will place a pointer in $A0-A1 to the descriptor of the string. The descriptor is discussed on page 137 of the Applesoft manual; it is simply three bytes of memory, the first being the length of the string, and the second and third the actual pointer to the string. The string may lie anywhere in memory, even in program text such as A$ = "123". However, it's better to have all the strings in one place; thus they normally build from HIMEM down. The String-Defined Function routines take advantage of the Apple's disregard for the location of the string, and so move it around to a place where the Apple can be convinced that it is a line of input to be interpreted. This is why signals do become crossed when using these functions during immediate execution, although they function properly.

The following is a list of Apple ROM routines, and explanations of their functions, that should be helpful in explaining the program's operation further.

$B8-B9: The text pointer of Applesoft. It points to the next token of program text to be interpreted.

$B1: A subroutine which causes the text pointer to be advanced to the next token. The ASCII value of that token is transferred to the A-register.

$A-B-C: The entry point of the USR function to the monitor. From here the default values of this program would send it to $3C7.

$A0-A1: The pointer to the descriptor of the string in USR(X). It is located within the FAC which occupies $9D-A2.

$DFE3: A subroutine which finds a variable, which is pointed to by the text pointer, in the variable list of the Apple. If it finds it, then it puts the address in $83-84 and in the X and Y registers. If it does not find that variable, then it creates or dimensions one if necessary, and then initializes it.

$DD6C: Checks to see if the last variable was a string. If not, you get a TYPE MISMATCH error.

$83-84: Pointer to a variable found by the routine at $DFE3.

$200-2FF: The Apple input buffer. All inputs are put and interpreted here.

$85-86: Used as a destination pointer.

```
032E: 99 00 02                STA  INBUFF,Y
0331: D0 F4                   BNE  LOOP1
0333: F0 F2                   BEQ  LOOP1
0335: A0 04       TOKEN       LDY  #$04       INIT VARIABLE FOR APPLE ROUTINES.
0337: 84 13                   STY  $13
0339: A2 FF                   LDX  #$FF       AIM TXTPTR
033B: 86 B8                   STX  TXPTRL     AT START OF
033D: A9 01                   LDA  #$01       INPUT BUFFER.
033F: 85 B9                   STA  TXPTRH
0341: 20 6C D5                JSR  TKNIZE     TOKENIZE THE BUFFER.
0344: A0 00                   LDY  #$00       NOW RECHECK LENGTH
0346: B9 00 02    LOOP2       LDA  INBUFF,Y   OF STRING IN
0349: F0 03                   BEQ  MOVEIT     BUFFER.
034B: C8                      INY
034C: D0 F8                   BNE  LOOP2
034E: C8          MOVEIT      INY
034F: 98                      TYA             SAVE THE NEW
0350: 48                      PHA             LENGTH COUNT.
0351: 20 52 E4                JSR  GETSPA     MAKE ROOM FOR NEW STRING.
0354: 98                      TYA             NOW SAVE NEW
0355: A0 02                   LDY  #$02       ADDRESS AND
0357: 91 83                   STA  ($83),Y    LENGTH.
0359: 88                      DEY
035A: 8A                      TXA
035B: 91 83                   STA  ($83),Y
035D: 88                      DEY
035E: 68                      PLA
035F: 91 83                   STA  ($83),Y
0361: A0 02                   LDY  #$02       POINT TO STRING
0363: A2 00                   LDX  #$00       AND MOVE IT TO
0365: 20 E2 E5                JSR  MOVSTR     ITS NEW LOCATION.
0368: 68                      PLA             RESTORE TXTPTR...
0369: 85 B9                   STA  TXPTRH
036B: 68                      PLA
036C: 85 B8                   STA  TXPTRL
036E: 60                      RTS             ...AND RETURN TO BASIC
                  *
                  * &LIST ROUTINE
                  *
036F: C9 BC       LIST        CMP  #$BC       IS IT 'LIST'?
0371: F0 03                   BEQ  LIST1      YES, GO TO LIST ROUTINE.
0373: 4C C9 DE                JMP  SYNERR     NO, PRINT 'SYNTAX ERR'.
0376: 20 B1 00    LIST1       JSR  CHRGET     GOBBLE FIRST CHAR.
0379: 20 E3 DF                JSR  PTRGET     FIND VAR. AND SAVE ITS ADDR.
037C: 20 6C DD                JSR  CHKSTR     MAKE SURE IT'S A STRING.
037F: A0 02                   LDY  #$02
0381: B1 83                   LDA  ($83),Y    GET STRING ADDRHI
0383: 85 86                   STA  $86        AND SAVE IT.
0385: 88                      DEY
0386: B1 83                   LDA  ($83),Y    LIKEWISE STRING ADDRLO...
0388: 85 85                   STA  REGL
038A: 88                      DEY
038B: B1 83                   LDA  ($83),Y    ...AND THE STRING LENGTH
038D: C9 FC                   CMP  #$FC       IS IT LONGER THAN 251 CHAR ?
038F: 90 05                   BCC  FAKELN     NO, OK.
```

```
0391: A2 B0                LDX  #$B0       YES. LOAD 'TOO LONG' TOKEN
0393: 4C 12 D4             JMP  ERROR      AND COMPLAIN.
                 *
                 * SET UP A PHONY PROGRAM LINE
                 * IN THE INPUT BUFFER
                 *
0396: A8         FAKELN    TAY
0397: A9 02                LDA  #$02
0399: 8D 00 02             STA  INBUFF     NEXT LINE ADDRLO
039C: 8D 01 02             STA  INBUFF+1   NEXT LINE ADDRHI
039F: 85 9C                STA  LOWTRH     STRING DESTHI IN LINE
03A1: A9 00                LDA  #$00
03A3: 8D 02 02             STA  INBUFF+2   SIMULATE END-OF-PROGRAM
03A6: 8D 03 02             STA  INBUFF+3   AT NEXT LINE.
03A9: 85 9B                STA  $9B        STRING DESTLO IN LINE.
03AB: 99 04 02             STA  INBUFF+4,Y BE SURE LAST CHAR IS $00.
03AE: 88         LOOP3     DEY
03AF: C0 FF                CPY  #$FF       COPY DONE?
03B1: F0 09                BEQ  READLN     YES, EXIT.
03B3: B1 85                LDA  (REGL),Y   NO, GET NEXT STRING CHAR
03B5: 99 04 02             STA  INBUFF+4,Y AND MOVE IT TO BUFFER.
03B8: D0 F4                BNE  LOOP3      DO IT AGAIN.
03BA: F0 F2                BEQ  LOOP3
                 *
                 * NOW FEED APPLESOFT THE LINE
                 *
03BC: A9 04      READLN    LDA  #$04       SET HTAB5
03BE: 85 24                STA  CH
03C0: A0 03                LDY  #$03       MAKE APPLE SKIP
03C2: 84 85                STY  REGL       LINE #.
03C4: 4C FC D6             JMP  LISTPR     APPLESOFT 'LIST' ROUTINE.
03C7: 20 6C DD             JSR  CHKSTR     MAKE SURE IT'S A STRING.
03CA: A5 B8                LDA  TXTPRL     FETCH TXTPTR
03CC: 48                   PHA             AND SAVE IT
03CD: A5 B9                LDA  TXPTRH     FOR LATER.
03CF: 48                   PHA
03D0: A0 02                LDY  #$02       STRING ADDRHI OFFSET.
03D2: B1 A0                LDA  ($A0),Y    AIM TXTPTR
03D4: 85 B9                STA  TXPTRH     AT STRING.
03D6: 88                   DEY
03D7: B1 A0                LDA  ($A0),Y
03D9: 85 B8                STA  TXPTRL
03DB: 20 67 DD             JSR  FRMNUM     EVALUATE AND PUT RESULT IN FAC
03DE: 68                   PLA             RESTORE TXTPTR...
03DF: 85 B9                STA  TXPTRH
03E1: 68                   PLA
03E2: 85 B8                STA  TXPTRL
03E4: 60                   RTS             ...AND RETURN TO BASIC


--- END ASSEMBLY ---

TOTAL ERRORS: 0

229 BYTES GENERATED THIS ASSEMBLY
```

$D56C: The main Apple interpreter routine. It will interpret and tokenize all characters in $200 until it finds a $0 which tells it to stop.

$E452: A routine which makes room for a new string in memory, which has its length in the A register. The A register returns unchanged, while the X and Y registers and $71-72 contain the starting address for the space for the string.

$E5E2: A routine that moves the string pointed to by the X and Y registers to the memory location pointed to by $71-72.

$71-72: Memory starting address for a new string; set up by the routine at $E452.

$DEC9: Do a SYNTAX ERROR and stop.

$D412: A routine which prints out an error message according to the error code value of the X-register. See page 136 of the Applesoft manual.

$9B-9C: Used as a destination pointer.

$D6FC: Address to a vector in the middle of the Apple LIST routine.

$DD67: A routine which evaluates the formula pointed to by the text pointer and puts its result in the FAC. The text pointer must be pointing at an expression that can be evaluated, and not at a string.

Using String-Defined Functions in Your Programs

There are basically two methods for using the String-Defined Functions routine in your own programs. The first is to poke it into memory using a subroutine in your program like that in the demo program. The second is to set up the routine in memory just once, save it to tape or disk in its binary form, and then load it directly into memory when needed. The latter method is particularly simple for disk users, and has the advantage of saving program memory space.

Method 1: POKEing from Applesoft. The subroutine in lines 50000-50060 can simply be transplanted into your own program, and initialized with a GOSUB 50000 somewhere near the beginning of your program. You are then free to use the &DEF, &LIST, and USR statements as described above.

Method 2: Loading the routine as a binary file. After typing in and running lines 50000-50060, the Machine Language routine can be saved directly to disk. From the keyboard, type in BSAVE FUNCTIONS,A768,L230 and the binary file will be created on your disk. Any Applesoft program can then simply BLOAD FUNCTIONS and do the few POKEs in line 50000 to set up the routine for use. Deleting lines 50010-50060, and substituting the line

50010 PRINT CHR$(4);"BLOAD FUNCTIONS": RETURN

will do the trick. If you want the routine installed at a different address than 768, simply specify that in the BLOAD command (e.g., BLOAD FUNCTIONS,A32768) and change the value of AD in line 50000.

The binary file can also be saved to tape. Instead of using the DOS BSAVE command, you would instead enter the monitor with a CALL -151, and then type in the monitor command 300.3E5W which will write the proper memory contents to tape. The routine can then be read into memory at any time from the same tape by using the monitor command 300.3E5R, and then doing the POKEs contained in line 50000 from your Applesoft program.

```
  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
  $      APPLESOFT BASIC       $
  $ '$-DEFINED FUNCTIONS DEMO' $
  $  AUTHOR: MICHAEL PRESCOTT  $
  $    (C) 1982    SOFTSIDE    $
  $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

100  TEXT : HOME : NORMAL : INVERSE
     : PRINT "*********************
     ************************" TAB(
     40)"**" TAB( 10)"$ DEFINED F
     UNCTIONS" TAB( 40)"**" TAB(
     40)"**" TAB( 16)"APPLE ]" CHR$
     (91) TAB( 40)"**" TAB( 40)"*
     ******************************
     ***************": NORMAL
```

Initialize the Machine Language routine.

```
110  GOSUB 50000
```

Input a string containing a mathematical/logical expression in terms of two variables. (There could be any desired number of variables.)

```
120  PRINT "TYPE IN A MATHEMATICA
     L/LOGICAL EXPRES-  SION IN T
     ERMS OF TWO VARIABLES X AND
     Y, USING NORMAL APPLESOFT SY
     NTAX:"
130  PRINT : INPUT F$
```

Call the Machine Language routine to convert the input string into a form usable by Applesoft as a defined function.

```
140  & DEF F$
```

Input values for the variables.

```
150  PRINT : PRINT "NOW ENTER VAL
     UES FOR X AND Y: ";: INPUT X
     ,Y
```

Invoke the defined function using a USR call.

```
160 Z =  USR (F$)
```

Print out the results, listing the defined function itself with the &LIST statement.

```
170  PRINT : PRINT "FOR X = ";X;"
      AND Y = ";Y;", THE FUNCTION
     ": PRINT
180  & LIST F$
190  PRINT : PRINT "HAS A VALUE O
     F ";Z
```

Pause and then go back for another function.

```
200  FOR I = 1 TO 3000: NEXT I: PRINT
     : PRINT : GOTO 120
```

Subroutine to POKE the Machine Language routine into memory, starting at address 768 (decimal). The starting address can be changed to another convenient location simply by changing the value of the variable AD in line 50000. This subroutine can be transplanted to another program, and need be executed only once, before using the &DEF or &LIST or USR statements.

```
50000 AD = 768: POKE 1013,76: POKE
      1014,AD - 256 *  INT (AD / 2
      56): POKE 1015,AD / 256: POKE
      10,76: POKE 11,(AD + 199) -
      256 *  INT ((AD + 199) / 256
      ): POKE 12,(AD + 199) / 256
50010  FOR X = 1 TO 5: READ A$: FOR
      Y = 1 TO  LEN (A$) STEP 2: POKE
      AD, ASC ( MID$ (A$,Y,1)) - 4
      8 + 10 * ( ASC ( MID$ (A$,Y +
      1,1)) - 48):AD = AD + 1: NEXT
      : NEXT : RETURN
50020  DATA "1D4B8D7:237A00237F3F
      238:1F5@4B275@5B270@207A1=3=
      4=6=7A1=3=3=6=7A1=8@9@003?00
      206=2C5I0H907A3=3?00208D"
50030  DATA "4H0H2H0@402=912@5I4=
      4B9@103=5B238:3E0@005B00200H
      300D8D8H0D2?27232B8F2?0@205>
      1=6=8=5>1=6=4:5>1=0@202@"
50040  DATA "00236F9F4:3=5B4:3=4B
      691D8B0H30671D2F237A00237F3F
      238:1F0@207A1=3=4=6=7A1=3=3=
      6=7A1=1D2I4>502@6A67812E"
50050  DATA "8@9@201>00201>10203=
      6?9@001>20201>30203=5?3?4020
      6=2C5I0H907A3=3?40208D4H0H2H
      9@403=630@302=3=672I4E23"
50060  DATA "8:1F5@4B275@5B270@20
      7A0@3=5B6=7A0@3=4B233:1F4:3=
      5B4:3=4B690A"
```

APPLE SWAT TABLE FOR:
STRING-DEFINED FUNCTIONS DEMO

LINES	SWAT CODE	LENGTH
100 - 200	ZE	502
50000 - 50040	YL	522
50050 - 50060	SS	176

Apple Diskourse

Part One

by Carey W. Bradley

This is the first in a series of articles dealing with the Apple Disk II system running under DOS 3.3, the Apple II's current Disk Operating System. The series is intended to help you gain a deeper understanding of how DOS 3.3 works, and to improve your skills in using the system. Many of the techniques we will discuss will be easily adaptable to systems operating under DOS 3.2, but it is preferable (and not just for the purposes of this series) that you upgrade your DOS 3.2 system to DOS 3.3 if at all possible.

The series will not be an elementary tutorial intended to replace the basic information presented in the DOS manual. Instead, we will assume that you've already read enough of the manual to be familiar with the fundamentals of DOS. Using the manual frequently as a reference, we'll expand on the material presented there, and look at specific examples of the DOS features and commands in use.

Nor will the series be a highly technical study, which would require a background in Machine Language programming to understand. BASIC is the common ground for ***SoftSide*** readers, and we'll stick to Applesoft throughout the series, with the assumption that your Apple has Applesoft in ROM. If your Apple has Integer BASIC in ROM, or you use a RAM-based Applesoft BASIC, you should still be able to adapt the information we present without much difficulty.

Each of the articles in the series will include Applesoft programs which illustrate and use the principles being discussed. This should allow you not only to make better use of DOS, but also to develop a respectable library of utilities you'll use over and over again. We will avoid using Machine Language routines whenever possible, in order to make the programs and principles understandable to all readers. Those who are fluent in Machine Language may wish to replace parts of our programs with their own routines for the sake of execution speed. Several excellent references are available if you choose to take this more technical approach.

Because of the nature of the procedures we'll be discussing, I'm obliged to give you the following warning, which I'll repeat throughout the series, at the risk of being redundant: When typing, saving, and (especially) testing the programs presented in this series, PLEASE use a new disk, or one that you have backed up. For all its power to perform apparent miracles, DOS also has the power to destroy, and an innocent typographical error could easily clobber valuable files on your disk. I recommend that you transfer your versions of my programs to a "good" disk only after you have thoroughly tested them on an expendable disk and are absolutely certain they do what they're intended to do.

Since this series is about the Apple disk system, you may find it odd that I have chosen to devote this first article not to drives or tracks or sectors, but rather to the Apple's memory — specifically, RAM. It is easy for most beginning programmers to just program away, without much regard for what is happening where inside the Apple. Even in programs using DOS commands this can often be done without unpleasant consequences, but if you do much of this type of programming, you'll come to realize how important it is to be aware of your program's use of RAM space.

Such a need could arise when you write programs large enough to tax your Apple's memory space limits, or when you use features such as high-resolution graphics, which require certain areas of memory to be reserved for special purposes. In fact, every time you boot DOS you're using a feature (DOS itself) which selfishly sets aside 10.5 of your K's for its own use. (Actually, this set-aside can be increased or decreased to some extent, but that's for another article). Knowing how your Apple is using its RAM not only allows for more efficient use of that space; it also helps to prevent or correct the inadvertent use of the same section of RAM for two or more different purposes. Such overlapping can give rise to assorted surprises, ranging from lost data and programs, to commands that don't work, to other things even too horrible to imagine.

Equally important is the fact that judicious use of RAM makes your programs more valuable. Suppose, for example, that you wrote a program on a 48K Apple which leaves more than 16384 (16K) bytes of RAM unused. If you were able to arrange it so that the program forced the unused memory to be the highest numbered RAM addresses, that program would probably execute just as well on an Apple with as little as 32K of RAM (there are exceptions). If so, when you submitted the program to your favorite software magazine it could be used by a greater number of Apple owners than if you had ignored RAM allocation altogether.

It isn't difficult to figure out how much RAM your program requires. It also isn't usually necessary that you figure it down to the exact byte, especially since things like string variables can cause the amount of RAM your program needs to fluctuate. If you go in for rules of thumb, try estimating your program's RAM requirements a little on the high side.

To start, determine the number of the highest RAM address available to your program. With DOS loaded at the top end of RAM, this address will range from 5632 on a 16K Apple to 38400 (also expressed as

-27136, which is 38400 minus 65536) on a 48K system. The exact values for all configurations are in Appendix D of the DOS manual. Then subtract 2048 (if Applesoft is in ROM), to account for the addresses 0-2048, which are mostly reserved for system use. The result is the total number of bytes at your disposal. Your BASIC program will be stored beginning in location 2049, and will reach upward in RAM.

The length of the program itself is easily found, because the address of its end is automatically stored in memory locations 175-176 when you load or enter it. With the program in memory, type

```
PRINT PEEK(176) * 256 +
PEEK(175) - 2049
```

and your Apple will tell you how many bytes long your program is. If this number is greater than 6142, the program reaches into the part of RAM reserved for high-resolution graphics page 1. If you try to use HGR in the program you'll find that the end of the program gets lopped off. You can still use HGR2 if your program length doesn't exceed 14334.

You may prefer to simply find the ending address of your program, by omitting the subtraction of 2049 from the statement shown above. You'll find it helpful to do this if your Applesoft is not in ROM; the above procedure assumes it is. The address where your program begins can, in any event, be found by typing

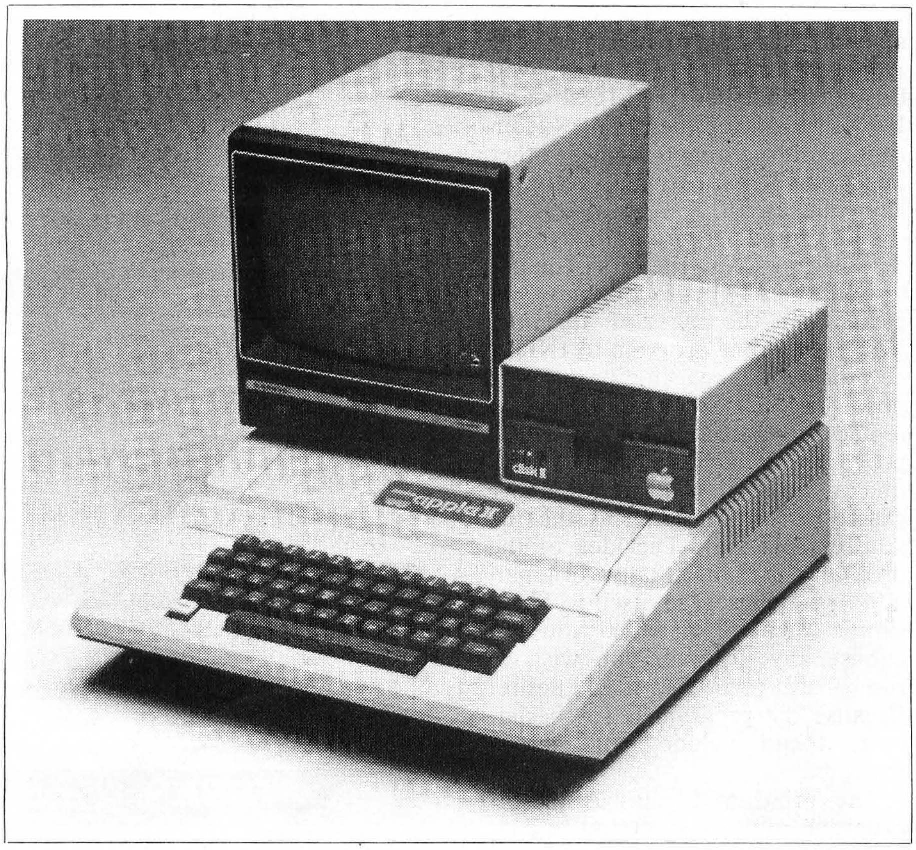
```
PRINT PEEK(104) * 256 +
PEEK(103)
```

All that remains is to determine how much space is needed for the program's variables, and where that space will be taken. Appendix D of the *Applesoft Reference Manual* tells you how many bytes are needed for each of the various types of variables. If the program uses Hi-Res graphics, you'll have to take into account the fact that variable names and numeric values are stored immediately after the end of your program (unless you have changed LOMEM), while strings are stored beginning at HIMEM and reaching downward. Again, your Applesoft manual will show you how to adjust HIMEM and LOMEM to keep these ranges from running into graphics areas or other important RAM segments you may have allocated.

This month's program is one designed to set up the disks you use to hold the Applesoft programs you glean from the pages of ***SoftSide***. You can use the program to INIT a new disk you'll be using for this purpose (see DOS manual), or to replace the greeting, or HELLO, program already on such a disk (make sure that you store it under exactly the same name as the file you're replacing). The idea of the program is not totally original, although the program itself is. It is a simple menu from which you can choose any program you wish to run, or exit to BASIC if you desire. Because the program is so specialized, it can include some unique features.

The program locates Applesoft programs in the disk catalog, and ignores all other types of files. To do this, it bypasses the DOS CATALOG command, and reads directly from the portion of the disk that contains the catalog information. This requires that you call the RWTS subroutine, which you'll have to accept on faith for now (or read that section of the DOS manual on your own). We'll discuss RWTS in more detail next time.

By avoiding the CATALOG command, you also avoid one of the problems of using it to generate a menu; a problem which arises when you have a large number of programs on one disk. CATALOG lists only 18 files, then awaits a keypress before continuing, at which point the first files listed may become unavailable to your menu, because they scroll off the screen as new files appear. This program reads the entire catalog into RAM, from there you can view the names of all Applesoft program files, a screenful at a time. You can page through this listing as many times as you like, without accessing the disk again. The upper limit on the number of files this program will handle is 105, far more than you'll probably ever have on a single disk. But DOS 3.3



allocates space for that many files in its catalog, so this program will handle any conceivable situation.

The program provides a good illustration of RAM allocation. It is written to run on the smallest RAM configuration it can, which is 20K. In the DOS manual it says that the highest RAM location available on a 20K Apple, without wandering into DOS territory, is 9728. If everything in the program is done below this address, DOS remains intact and operational as long as the Apple running the program has 20K or more of RAM. I need 3840 bytes to store the catalog information, so I lower HIMEM to 9728-3840 = 5888. When I subtract out my 2048 "system" bytes, I am left with 3840 bytes for program and variable storage. If you enter the program exactly as I have, you can determine its length to be 1112 bytes. The variables use less than 350 bytes. The data for the RWTS subroutine are tucked away in a handy area below the RAM space I'm considering, so they don't count in my calculations. This leaves me nearly 2400 unused bytes; a good margin, but far less than the 4096 I'd need to be able to run the program as it is currently written with only 16K of RAM.

The operation of the program is described step by step in the documentation which is spliced into the listing. The variables used are also listed and described. If you like the way the program works, you can return to it when you're finished running another program. Just locate the line at which the program ends, and insert

```
PRINT CHR$(4);"RUN HELLO"
```

Or, if your program has already defined the variable D$ to be CTRL-D,

```
PRINT D$;"RUN HELLO"
```

It is a good programming practice to reserve the variable name D$ for this purpose, in ALL programs. There are plenty of other names you can use for other things. After you revise the program, be sure to save it back to the disk again in its new form.

You can change all of the programs on the disk to do this, even if some are the kind that you have to end yourself by hitting CTRL-C. In these programs, an ONERR GOTO statement can be used to branch to one of the above statements when you interrupt the program.

In future articles, we'll discuss the uses of DOS commands inside and outside programs, what happens when you use them, what happens when you misuse them, and how to use them better. I've got plenty of my own ideas on what we should cover, but I'd be happy to interrupt my sequence to deal with topics you suggest. Please let me hear from you, c/o ***SoftSide***.

Next time, we'll get out of our RAM and back on track — and sector. We'll investigate the ways in which DOS stores information on the disk, and take a closer look at the RWTS subroutine and how you can use it to do some useful tricks, even if you're not familiar with Machine Language. In subsequent articles, we'll make frequent use of the RWTS subroutine, both in homebrewed disk utilities and as a microscope for examining what really happens when we store information on a floppy disk using Apple DOS.

Catalog Program Variables

I, J, L: Counters and loop variables.
A%(*): Array used to store the starting RAM addresses of the names of Applesoft programs on the disk.
BF: Location of the byte used to determine where the information being read from disk will be stored in RAM.
D$: The conventional variable for CTRL-D, used to prefix DOS commands.
IN$: Your input.
N$: The name of the program you have selected to run.
NF: Counter for the number of Applesoft programs on the disk.
RW: Starting address of the RWTS soubroutine data.
S: Address of the byte which tells RWTS which sector to read. (All data are read from the same track, which is already included in the DATA statement).

APPLE™

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$ 'DISKOURSE HELLO PROGRAM'    $
$   AUTHOR: CAREY BRADLEY      $
$    (C) 1982    SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Save the original value of HIMEM in two unused zero-page memory locations. (This value is restored before exiting the program, through either line 320 or line 360.

```
100  POKE 254, PEEK (115): POKE 2
     55, PEEK (116)
```

Set HIMEM, as described in the article, and initialize a few things.

```
110  HIMEM: 5888:BF = 786:RW = 76
     8:S = 782
120  TEXT : HOME :D$ =  CHR$ (4):
      DIM A%(105)
```

Read the data required by the RWTS subroutine.

```
130  FOR I = RW TO RW + 29: READ
      J: POKE I,J: NEXT
```

Use the RWTS subroutine to read the disk catalog into the 3840 bytes of RAM reserved for it.

```
140  FOR I = 15 TO 1 STEP  - 1
150  POKE S,I: POKE BF,38 - I
160  CALL RW: NEXT
```

Search the catalog data for Applesoft programs, and remember the memory address at which each of these program names begins.

```
170  FOR I = 5900 TO 9484 STEP 25
     6
180  FOR J = I TO I + 210 STEP 35
190  IF  PEEK (J) = 255 THEN 220
200  IF  PEEK (J + 2) <  > 2 AND
      PEEK (J + 2) <  > 130 THEN
     220
210 NF = NF + 1:A%(NF) = J + 3
220  NEXT J,I:J = 1
```

Display the program names and prompt the user to select the action to be taken. Lines 320 and 330 provide the two ways to end the program — by exiting to BASIC or by running one of the programs. Note that both of these lines call the subroutine in line 370 before proceeding, which undoes what line 100 did, giving you back the amount of memory you had available before running the program.

```
230  HOME : INVERSE : PRINT "DISK
      VOLUME "; PEEK (791): PRINT
     : IF J > NF THEN J = 1
240  PRINT "APPLESOFT PROGRAMS FR
     OM 'SOFTSIDE':": NORMAL : PRINT
     :L = 1
250  IF J <  = NF THEN  PRINT "<"
     ;J;">"; TAB( 7);
260  FOR I = A%(J) TO A%(J) + 29:
      PRINT  CHR$ ( PEEK (I));: NEXT
     : PRINT
270 L = L + 1:J = J + 1: IF L > 1
     6 OR J > NF THEN 290
280  GOTO 250
290  VTAB 22: INVERSE : PRINT "EN
     TER PROGRAM NUMBER TO RUN"
300  IF NF > 16 THEN  PRINT "ENTE
     R 'M' TO CONTINUE LISTING"
310  INPUT "OR HIT <RETURN> TO EX
     IT TO BASIC:";IN$: NORMAL
320  IF IN$ = "" THEN  GOSUB 370:
      END
330  IF IN$ = "M" THEN 230
340 I =  VAL (IN$): IF I < 1 OR I
      > NF THEN 290
350  FOR J = A%(I) TO A%(I) + 29:
     N$ = N$ +  CHR$ ( PEEK (J)):
      NEXT
360  GOSUB 370: PRINT D$;"RUN";N$
370  POKE 115, PEEK (254): POKE 1
     16, PEEK (255): RETURN
380  DATA  169,3,160,9,32,217,3,9
     6,0,1,96,1,0,17,0,26,3,1,0,0
     ,0,1,0,0,96,1,0,1,239,216
```

Swashbuckler

A review by Alan J. Zett

by Paul Stephenson. From DATAMOST, Inc., 9748 Cozycroft Ave., Chatsworth, CA 91311, (213) 709-1202. System requirements: 48K APPLE II/II+ and disk drive with 16-sector controller. Suggested retail price: $34.95

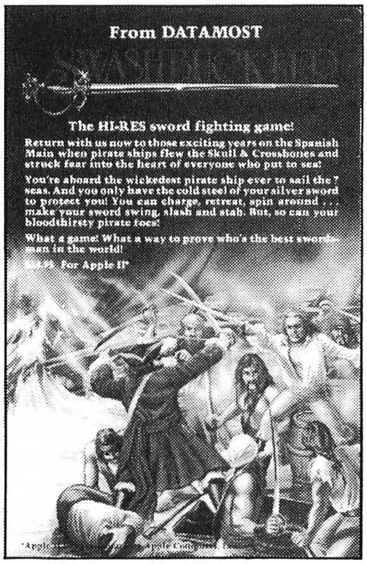
When was the last time you ventured into the realm of swordfighters, pirates, and cutthroats? If you're a fan of the *DUNGEONS & DRAGONS*™ games, chances are that you do so on a regular basis. Or perhaps you are one of the many thousands of people who enjoy the Science Fiction/Fantasy or Sword and Sorcery book genre. Many times in my youth I spent hours in high adventure, fighting imaginary battles with monsters, demons, and sorcerers, courtesy of the creative genius of authors such as Robert E. Howard and Edgar Rice Burroughs; but, in all my conquests, I never once had the chance to actually participate in a real swordfight.

Now DATAMOST has come out with their exciting program called *Swashbuckler*. Your goal is to escape from the hold of the pirate ship where you are being held captive, or to take as many of your captors as possible down with you. All you are armed with is a sword and the determination to make it to freedom alive. Above you lies a ship full of professional fighters, each with a different taste in weapons and each with a personal fighting style.

Both you and your opponents (controlled by the computer) must skillfully dodge, thrust, and parry to stay alive. Points are scored according to the number of opponents you have slain. You are allowed three deaths, but if you make it to 25 points you will gain an extra life.

The fighting is fast-paced, realistic, and beautifully drawn. When you first start out, you are given one opponent. After a short while you are given two, one on either side. When you've gained enough experience to overcome two opponents, the computer starts making things harder. First your opponents get tougher to beat. They gradually get quicker with their swordplay, and gain an incredible knack for offensive/defensive maneuvering. To make things even more difficult, when enough time has passed, your opponents change. Since each fighter has a personal technique, different combinations of opponents can make the game a never-ending struggle which tests your fighting skills to their utmost.

Fighters range in skill and style from a boisterous gargantuan with a club, to pirates, to soldiers in full plate armor. Each fighter uses a different weapon. There are knives and axes, clubs, cutlasses, javelins, pikes, and broadswords. As if this weren't enough, certain obstacles such as scurrying rats and snakes will plague you from time to time. Occasionally a dead body will block your retreat. If you make it to the big time, the obstacles become more difficult. Giant spiders and scorpions attack you. If you're bitten by a rat, you lose the ability to thrust; but if a scorpion bites — you're dead!



The HI-RES sword fighting game!

Return with us now to those exciting years on the Spanish Main when pirate ships flew the Skull & Crossbones and struck fear into the heart of everyone who put to sea!

You're aboard the wickedest pirate ship ever to sail the 7 seas. And you only have the cold steel of your silver sword to protect you! You can charge, retreat, spin around . . . make your sword swing, slash and stab. But, so can your bloodthirsty pirate foes!

What a game! What a way to prove who's the best swordsman in the world!

[illegible] For Apple II*

By the time you've reached the main deck, you have a multitude of little beasties snapping at your feet while the battle rages all around you. The scene is something straight out of a Conan novel; your back is against the wall and you are fighting desperately to stay alive.

All this action would be a bit overwhelming if it were not so well presented. The characters are all sharp and clear. The detail and animation is an achievement in itself. To back up the splendid detail of the fighters, a superior full-page display was needed. The author did not disappoint me here either. The display consists of a series of beautifully drawn, full color, Hi-Res pictures that serve as the backdrop for the scenarios. As you reach different levels of the ship, the picture changes to reflect your new location. This helps tremendously in setting the mood for the game. A small area is cleared in the middle of the screen for the fighters. It allows an area for you and your opponents to move around in.

Fighting is controlled from the keyboard. There are commands for facing left or right, advancing or retreating, on guard, parry, high parry, low parry, and thrust. The sword movement keys are grouped together in a diamond shape on the right side of the keyboard and have a natural pattern that is at once logical and easy to use. The movement keys are set in a line on the left side and likewise are in a very natural position for ease of use. Before long you'll be using the keyboard as if you were a master swordfighter.

As you can see from my enthusiasm, I find *Swashbuckler* to be an extremely addicting, well conceived program devoid of bugs and annoyances. *Swashbuckler* is an excellent, original Machine Language game, a welcome addition to the commercial software market. I highly recommend this program to the adventure-seeker in all of us. ⑤

"*DUNGEONS & DRAGONS*™ is a trademark owned by and used under license from TSR Hobbies, Inc."

Apple-Aids

A review by Carey W. Bradley

By Allen Wyatt (Advanced Operating Systems). System requirements: 48K Apple II or Apple II Plus with Applesoft in ROM or language card, one or more disk drives, DOS 3.2 or 3.3. Suggested retail price: Disk and manual, $49.95

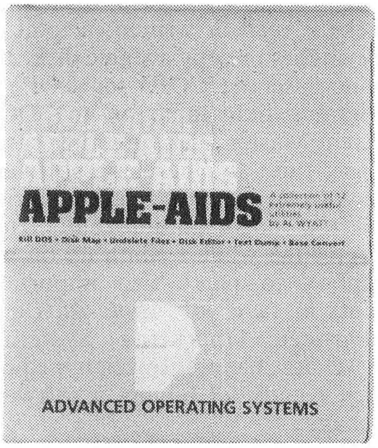
The nice thing about Apple DOS is that anyone who can talk to an Apple in BASIC can easily learn to use it. With a minimum of effort, the fundamentals of SAVEing, LOADing, RUNning, LOCKing, DELETEing, and so on, can be quickly mastered. With a little more effort you can effectively use sequential and random access text files in your programs.

But as your programming efforts become more ambitious, it is inevitable that you'll need to know something about what's actually going on when your DOS commands are executed. *Apple-Aids* is a handy package of twelve utilities designed to give you more intimate access to the data on your disks than the regular DOS commands allow. With these programs you can examine and manipulate any of the information on your normally formatted DOS 3.2 or 3.3 disks.

I was impressed by the combination of power and simplicity I found throughout the package. Everything in it is straightforward and (forgive the expression) polished. The screen displays are informative and easy to understand. Even the system's main menu is cleverly conceived, and I must confess that I spent some time playing around with it before I even tried running any of the utilities.

If documentation is to be judged on the basis of both form and content, as I believe it should be, *Apple-Aids* makes the grade on both counts. Packaged in a sturdy, stand-up, loose-leaf binder, the manual is typeset, printed on high-quality paper, and is well illustrated. The text is well-organized and indexed. It is clear, complete and concise, with no fluff. Use of the *Apple-Aids* package requires some knowledge of the way DOS stores information on the disk. If you lack this knowledge; fear not, the manual includes all of the basic information you will need. What's more, you can learn to use *Apple-Aids* in a single sitting.

The most powerful of the *Apple-Aids* functions is the "Disk Editor." With it you can access any sector on a disk, and see its contents displayed in decimal, hexadecimal, character or disassembled form. A condensed display mode shows the entire sector in both hexadecimal and character form simultaneously. While the contents of the sector are in the Apple's memory, you can edit the data and write it back to the sector of your choice. You have complete control of these functions; editing can be done in decimal, hexadecimal or character mode, and you can easily select any slot, drive, track and sector you desire for any read/write operation.



Sector editing is dependent, of course, upon knowing what sector you're looking for. *Apple-Aids'* "Sector Listing" option makes this a snap. Each file in the disk catalog is listed along with the tracks and sectors which contain its data. An "Improved Directory" feature lets you look at the disk catalog in a different way. This option lists all files in the catalog, including ones which have been deleted, showing the file's name, type, and status (locked, unlocked or deleted). For each binary file, the address at which it will be BLOADed and its length are also shown.

When you need a more general view of sector usage on a disk, bring up the "Disk Map" utility. On your screen you will see a table of 35 columns and 16 rows (13 for DOS 3.2), with an asterisk marking each sector currently in use: a graphic representation of what DOS looks at when it decides which sectors are available for receiving new data.

If you accidentally DELETE a file, you can use the "Undelete Files" utility to bring it back to life. This is done by restoring the original file type to the catalog entry and altering the Volume Table of Contents to show that the appropriate sectors are in use. The success of this option depends on whether you have written over any of the file's sectors in the meantime; but if you use it right after you make the deletion it can be a lifesaver. (If data has actually been erased by a DOS operation, you can really mess up a disk, so heed the warning given in the manual.)

On a normally INITed disk, 31 of the 35 tracks are available for your use. Three are reserved for DOS and one for catalog information. Two *Apple-Aids* options allow you to increase the available storage space by creating a disk without DOS. The "Format Blank Disk" utility, as its name implies, gives you a blank, formatted disk without DOS. This is useful for creating disks which will be used only for data storage; DOS will already (presumably) be contained on your program disk. "Kill DOS" is similar, but it just removes DOS from the disk, without altering any other data already stored on the disk. Actually, only two additional tracks are made available to you by either of these methods, so you get 32 or 26 sectors worth of space, depending upon which version of DOS you are using. Of course, the disk can't be booted, but you won't care if your application requires the additional disk space. A nice feature of these options is that they store information on track 0 to notify you if you mistakenly attempt to boot the disk. This is preferable to sitting spellbound as your disk drive runs on and on.

EXEC files are a nice feature of Apple DOS, but they can be a hassle to create and revise. *Apple-Aids* has just what you need if you do a lot of EXECing. "Create Exec Files" and "Edit Exec Files" do just what they say, taking most of the headaches out of your EXEC file work. (If you're like me, you'll probably still find ways to create some more.)

Rounding out the package is a useful, if somewhat slow, program which converts number bases, changing any decimal, binary, hexadecimal, or octal number to its equivalent in each of the other numbering systems. This doesn't actually qualify as a disk utility, but it's awfully nice to have it there to use in conjunction with the others.

My least favorite of the utilities is the "Disk Copy" function. Its advantage is that the video display shows you which track is currently being copied. This is overshadowed, however, by the disadvantage that it copies only one track at a time. Copying with two drives is just slow, but copying with a single drive is maddening. It requires that you switch disks after each of the 35 tracks is read or written, a total of 70 switches to copy a single disk. I'd sooner use the COPY or COPYA programs that came on my DOS System Master Disk, in either case.

Any time you purchase software for your Apple, you're taking a risk — it's unfortunate, but true. Forgive me while I climb up on my soapbox to make two important points in favor of *Apple-Aids* and Advanced Operating Systems. These are factors I weigh heavily when I consider buying a software package.

First, the most important element, support. I had a problem getting the *Apple-Aids* "Text Dump" option to work with my printer. Everyone said that they support their products, but I knew better. Prepared for the runaround I'd received in quests for solutions to software problems in the past, I dialed the AOS phone number I found in the manual. Wonder of wonders, I didn't get a recording telling me to ask my dealer about it, or a sales department where they know nothing more about a package than its stock number. I got, instead, a live person in the AOS office, so I decided to go for it, and asked for the program author. Another miracle — my call was put through to him, and in spite of what you might think, I didn't have to identify myself as a reviewer to receive this courtesy.

In short, the print routine was promptly rewritten to correct the problem. When I did tell Mr. Wyatt I was reviewing his package, he asked me to inform my readers that, in the event they got a copy of *Apple-Aids* that had been shipped before this problem was detected, they should keep a backup copy and return the original to AOS for a FREE replacement. They'll even pay the shipping costs, he said. Although this sounds as though it should be routine procedure throughout the industry, believe me, it isn't.

I mentioned that you can make a backup copy, which brings me to my second point: copy protection. *Apple-Aids* is not copy-protected. I would agree that, to some extent, software piracy is everyone's problem, but copy protection itself is often a problem; and when it is, it's nobody's problem but the user's. When I pay for a piece of software, I want the ability to immediately make a backup copy, without paying again. And if I somehow clobber my original, I want to be able to make another backup right away, in case I'm stupid enough to do the same thing again. I don't care for the fact that some software, especially games, is locked up tighter than a drum. I know from experience that the software I can use as a programmer is often useful in inverse proportion to the amount of copy protection it has.

I hope some of you software pushers are listening, because I am a paying customer, and I speak for others like me: I am much more likely to buy your product if I can legitimately copy it, and if I know that direct support is readily available. *Apple-Aids* has both of these features. I'll get off the soapbox now.

The longer you have *Apple-Aids*, the more uses you'll find for it. It can greatly enhance your programming and trouble-shooting capabilities in a DOS 3.2 or 3.3 environment. I not only recommend *Apple-Aids*, I also advise you to take a closer look when you see the Advanced Operating Systems name on other software packages. I know I will.

K-Byter

Sound Mixer

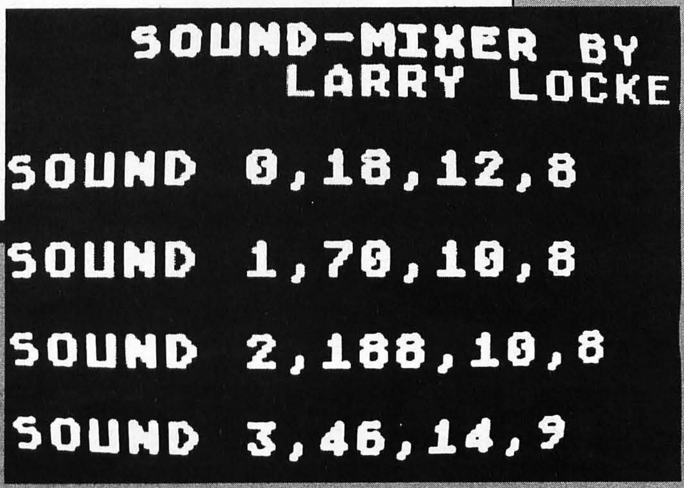
An ATARI® K-Byter by Larry Locke, Nazareth, PA

This is an excellent program to acquaint ATARI® users with the incredible variety of sounds that can be generated by their computers. It allows you to mix and modify all four sound voices in real time, using a joystick plugged into port #1.

Pressing the fire button chooses the voice to modify, as shown on the screen display in red. Moving the stick up or down increases or decreases the pitch. Moving to the left changes the distortion value, and moving to the right changes the volume level.

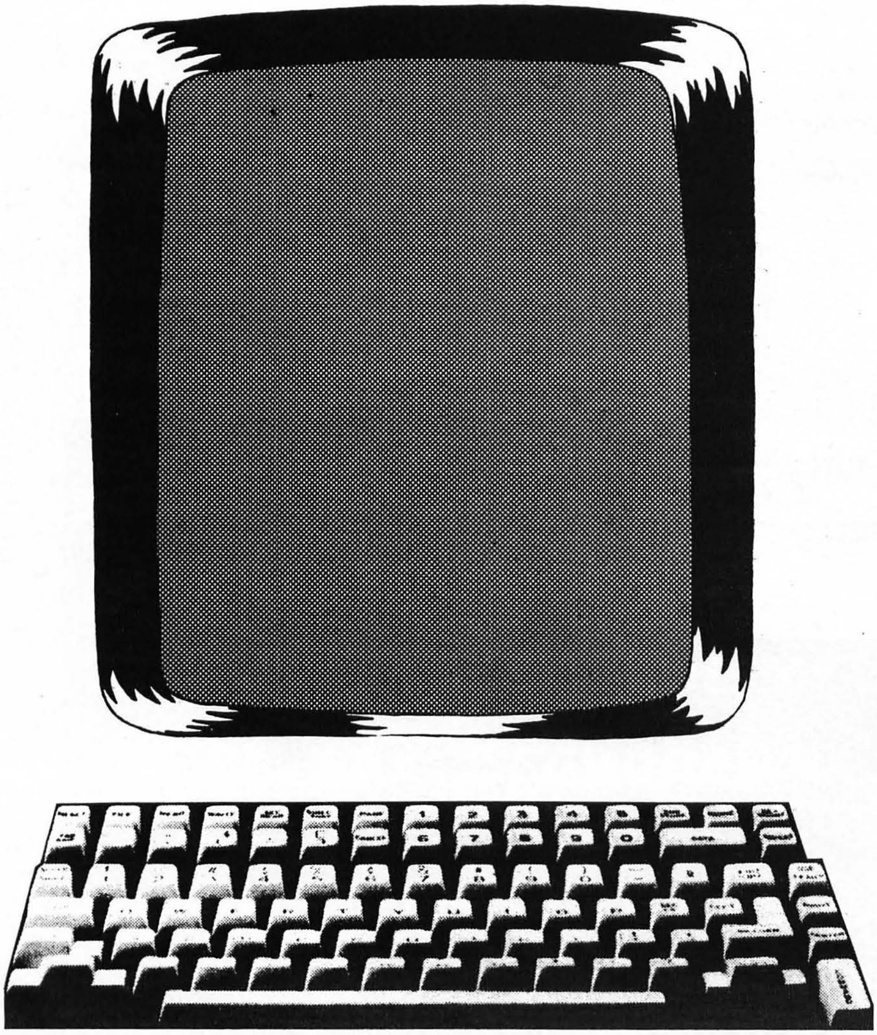
Have fun creating new harmonies and cacophonies!

```
10 DIM PITCH(4),DIST(4),VOL(4)
20 FOR A=1 TO 4:PITCH(A)=0:DIST(A)=10:
VOL(A)=8:NEXT A
30 GRAPHICS 18:? #6;"    SOUND-MIXER b
y"
40 ? #6;"          LARRY LOCKE":GOSUB 50
0
60 X=1:GOTO 1000
500 N=0:FOR S=3 TO 9 STEP 2:POSITION 1
,S:? #6;"SOUND ";N:N=N+1:NEXT S:RETURN
1000 IF STRIG(0)=0 THEN GOSUB 500:X=X+
1:GOTO 1055
1020 IF STICK(0)=13 THEN PITCH(X)=PITC
H(X)+1:IF PITCH(X)>255 THEN PITCH(X)=0
1030 IF STICK(0)=14 THEN PITCH(X)=PITC
H(X)-1:IF PITCH(X)<0 THEN PITCH(X)=255
1040 IF STICK(0)=11 THEN DIST(X)=DIST(
X)+2:FOR I=1 TO 30:NEXT I:IF DIST(X)>1
4 THEN DIST(X)=0
1050 IF STICK(0)=7 THEN VOL(X)=VOL(X)+
1:FOR I=1 TO 30:NEXT I:IF VOL(X)>15 TH
EN VOL(X)=0
1055 IF X>4 THEN X=1
1060 POSITION 1,X+X+1:? #6;"sound"
1070 POSITION 8,X+X+1:? #6;",";PITCH(X
);",";DIST(X);",";VOL(X);"  "
1080 SOUND X-1,PITCH(X),DIST(X),VOL(X)
:GOTO 1000
```



```
    SOUND-MIXER BY
          LARRY LOCKE
SOUND 0,18,12,8
SOUND 1,70,10,8
SOUND 2,188,10,8
SOUND 3,46,14,9
```

ATARI® RANDOM ACCESS DATABASE



by Paul Marentette

Random Access Database **is a database management program for an ATARI® 400/800, requiring 32K RAM and a disk drive running under DOS 2. It is included as a bonus program on this month's ATARI® Disk Version.**

Those of you who have a disk drive and ATARI®'s DOS 2 are in for a treat. This totally revised version of the database makes full use of the random access capabilities of disk storage.

Until now the database has been an "in-memory" system. All records were kept in the computer's memory at all times. If only one record was added or deleted from the database, the entire database had to be rewritten to the tape or disk. With a large database this is time consuming and inefficient.

In-memory databases are also limited in size because data must fit into available RAM. For example, if you had a 40K machine, you could have a database of approximately 170 90-byte records. With the random access database (and 480 free sectors on a disk) more than 770 of these records could fit in the database.

Before examining the new database, let's consider how ATARI® handles random disk access. The NOTE and POINT instructions are used to keep track of where each record resides on the disk. When a file is opened in mode 12 (OPEN #IOCB,12,0,"D:FILENAME") an internal pointer positions the disk's read/write head at the beginning of the file. Under program control the POINT #IOCB,Sector,Byte instruction is then used to reposition the disk head at any sector and byte in the file.

In this manner the computer can automatically calculate where a record resides on the disk, and bring only that record into memory. Only one complete record ever resides in RAM; this frees up the bulk of memory for storing the pointers (disk addresses) for each record in the database. Since each pointer uses only 3 bytes, a much larger database can be created.

The pointers, consisting of the sector (1-720) and byte (1-125) address of each record, are stored in the P$ string. They could be stored in a numeric array, but that would require 12 bytes of memory for each sector-byte pair. Instead, by using the CHR$ and ASC string functions, the two numbers can be stored in 3 bytes. The first two bytes store a sector number. Byte #1 contains the number of 256's in the sector number; byte #2 contains the remainder. And byte #3 contains the byte portion of the disk address. Thus an address of sector 550, byte 63 would have a pointer in the P$ string consisting of CHR$(2), CHR$(38), and CHR$(63).

The routine from line 1070 to 1110 writes blank records into the disk file and stores their NOTEd addresses in the P$ string using the CHR$ function. Lines 170 to 180 contain a reverse process which uses the ASC function to decode the P$ characters into byte and sector numbers and then point the disk head to the calculated address. The program then reads (line 1160) or writes (line 1140) a record at that address.

Stepping Through the Program

Although this version of the database follows the general outline of the earlier one (***Softside***, December, 1981), it is significantly different in many areas.

Lines 110 to 700 contain variable initialization, frequently used subroutines, and the main menu and disk directory read routines. Not much has changed here, but things have been moved around and some of the subroutines are new.

The really important changes have occurred in the load and save file routines and the new file initializing routine. Let's examine the last of these first. Starting at line 850 the number and names of headings are established. The numeric array B(NH + 1) is used to store the starting bytes of each field since we are now allowing fields to have different lengths. The earlier database padded every field with blanks so that all fields were the same length as the longest one — a terrible waste of space.

Throughout the program the S, E and L variables are used to specify the start, end, and length of a field within the total record. These values are recalculated any time the field in question changes.

Once the headings are known the program calculates the maximum number of records that can be accommodated, given the two variables of available RAM and disk space. Then the tricky business starts. We could set aside enough disk space for MX records; but, if the database is a small one, disk space will be unnecessarily dedicated to the database file. We could add a new record to the end of the file each time one is needed; but that is wasteful too — ATARI®'s File Management System grabs a new disk sector every time a record is added (using IO mode 9) and ignores any vacant bytes that might have existed on previously used sectors.

The way I solved this problem was to "block" records in groups that come as close as possible to filling complete sectors. Up to 50 records (an arbitrary number) can be in a block. An efficient block size between 1 and approximately 50 is calculated in lines 1000 to 1040, and then the subroutine at line 1070 is called to do the actual setting up of file space. The variable RA keeps track of total file space already set aside. When more space is needed, another block is added on.

Every database file you create will have two disk files associated with it: The one with a ".DAT" extension will hold the actual records, and the other with a ".HDG" extension will hold the heading names, file size information, and the P$ string containing the pointers to all records in the ".DAT" file. The heading file is saved in lines 1180 to 1260. Variables BLK and RA are saved along with the others so that the database program always knows how a particular file is blocked and what space is already set aside. The ".HDG" file will grow in size as the database grows; as more records are added, the pointer string gets longer, and that string is saved up to its current length.

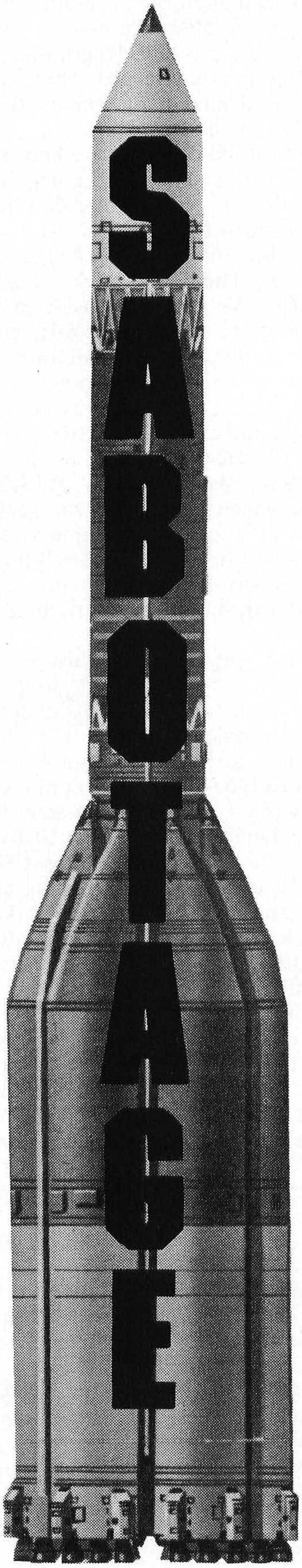
When an old file is loaded (lines 720 to 800), the ".HDG" file is read first and variables are *dimensioned* as needed. Then a routine at line 820 is called to read into memory one complete set of fields for the first heading. In other words, the first heading's contents for every record are placed in memory. The CH variable keeps track of which heading is the current one. If a sort or search is done on a different field, that field's contents for every record are read in using this same routine.

The add and change records routines share a common input data subroutine (lines 1710-1730). If, when a record is being added (line 1600), available file space (RA) has been filled, a new block of space is created by calling the routine at line 1070. The variable IO is set to 9 so that the file is opened in the append mode. In the change record section the variable CS is checked to see if any changes were made to the current record. If so (CS = 0), then the subroutine at line 1140 is called to rewrite the record at the same location on the disk from which it was read.

In the delete and sort records routines the real beauty of random access is seen in action. When a record is deleted from the database it is not removed from the disk. Instead, the pointer to that record's disk space is moved to the end of the P$ pointer string (line 1900) so that the deleted record's disk space can be made available for new records. In the sort routine, whenever the sort field is moved around, the corresponding pointers in the P$ string are moved around too. In effect, the records are never moved on the disk — only the pointers get sorted!

Data is automatically saved when you quit the program. (Don't exit the program any other way than the Q menu option!) If you want to save a copy of your file in the midst of working, the save option will save a copy of your current heading and pointer file. The data itself is always saved automatically, immediately after it is entered or changed.

Any other revisions, not described here, are merely adjustments in the coding to accomodate the new random access format. For help with using the program's features check Mark Pelczarski's article on the Apple version in ***SoftSide***, October, 1981.



by Peter Adams

***Sabotage* is an arcade-type game for an ATARI® 400/800 with 16K RAM and one joystick.**

Your mission is to destroy the enemy's main computer. You will travel from room to room (left to right) in order to gain access to it. In each room there is a laser that will relentlessly fire upon you.

As the game begins, you are represented by the orange dot in the upper left of the screen. The yellow-green dots scattered around the room are control boxes. The blue lines are walls and the yellow-green machine at the top center is the enemy's laser (surrounded by its shields). All items shown are highly electrified and will destroy you upon contact.

You are able to fire a missile in the direction you are moving, or, if you are stationary, in the direction you last moved. You may only have one missile in play at a time.

If you hit a control box with your missile, that control box is temporarily disabled (1 point). If you can disable six or more control boxes, the transporter on the right wall will allow you to pass into the next room. If you hit the right wall while there are five or more control boxes left, you will be transported back to the left side of the board. Leaving a room while all ten control boxes are disabled is worth five bonus points.

Before the enemy's laser fires at you, you will be warned by the dropping of its shields, while it locks onto your position at that time. Shortly it will fire a laser at that point; if you are still there, or you are in the line of fire, you will be destroyed. If you shoot the laser with a missile while its shields are lowered, it will be disabled (5 points). While the laser is disabled, no control boxes will regenerate. A laser can only be disabled once per room.

If you are able to reach the main computer (room five on your first mission), you destroy it by moving yourself into it. This sets off a chain reaction which destroys the whole complex. Your mission is then complete (30 points) and you must escape before it is too late (10 points).

You have three lives to complete your mission. If you succeed, you will be given another mission more difficult than the first.

Variables

B: Room number.
D: Lives left.
DX: Missile X change if fired.
DY: Missile Y change if fired.
G: Difficulty.
GT: GOTO variable.
H, I: Work variables.
L: Laser present (1) or gone (0).
M: Missile on (2) or off (0).
MX: Missile X coordinate.
MY: Missile Y coordinate.
SC: Score.
T: Work variable.
T1: Time counter for laser firing.
T2: Time spent in room.
T3: Time counter for increasing difficulty.
T4: Time counter for laser.
V1: Laser has (1) or has not (0) been fired in a room.
V2: Color in which to plot disappearing missile.
V4: Length of walls in escape room.
V5: Time counter for escape room.
V7: Frequency of invisible walls.
V8: Maximum difficulty.
V9: Mission number.
X: Your X coordinate.
XM: Missile X change.
X(10): Control boxes' X coordinates.
XC(15): X change for stick positions.
Y: Your Y coordinate.
YM: Missile Y change.
Y(10): Control boxes' Y coordinates.
YC(15): Y change for stick positions.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          ATARI BASIC         $
$          "SABOTAGE"          $
$      AUTHOR: PETER ADAMS     $
$       (C) 1982 SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization.

```
100 X=5:Y=10:L=1:D=3:MY=47:V9=1:GT=690
:V8=2:B=1:DX=1
120 DIM XC(15),YC(15),U$(1),T$(1):FOR
T=5 TO 15:READ H,I:XC(T)=H:YC(T)=I:NEX
T T:U$=CHR$(28):T$=CHR$(127)
130 DATA 1,1,1,-1,1,0,0,0,-1,1,-1,-1,-
1,0,0,0,0,1,0,-1,0,0
140 DIM X(10),Y(10)
```

Set up incomplete Player Missile Graphics.

```
150 POKE 53277,3:POKE 704,130:POKE 705
,216
```

Reset another room.

```
200 GRAPHICS 5:POKE 752,1:POKE 77,0:SE
TCOLOR 0,2,6:? "SABOTAGE",
210 ? T$;"ROOM ";(B-V9)+1;"  MISSION "
;V9:G=7-B:? "SCORE",,,"LIVES":? SC,,,D
:T2=0:V1=0:IF B=5 THEN 7000
```

Draw center wall for room four.

```
220 GOSUB 6000:IF B=4 THEN PLOT 39,40:
DRAWTO 39,9:DRAWTO 41,9:DRAWTO 41,40
```

Draw other walls.

```
230 FOR T=10 TO 60 STEP 20:PLOT T,2:DR
AWTO T,20*RND(1)+14:PLOT T+10,40:DRAWT
O T+10,31-20*RND(1):NEXT T:GOSUB 4100
```

Draw and store control box locations.

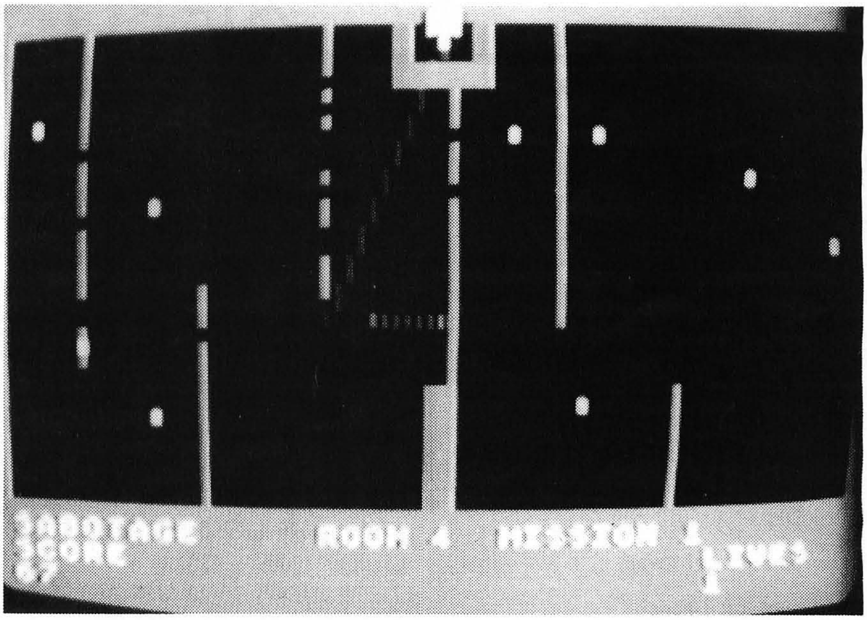
```
240 COLOR 2:FOR T=1 TO 10:H=INT(70*RND
(1)+5):I=INT(27*RND(1)+10):PLOT H,I:X(
T)=H:Y(T)=I:NEXT T:GOSUB 4000
```

Draw yellow walls for room three.

```
250 IF B=3 THEN PLOT 30,10:DRAWTO 30,2
0:PLOT 50,10:DRAWTO 50,20
```

Check joystick and change player position.

```
500 T=STICK(0)
540 COLOR 0:PLOT X,Y:X=X+XC(T):Y=Y+YC(
T):LOCATE X,Y,H:IF H>0 THEN 2000
```



Adjust missile direction.

```
560 IF T<>15 THEN DX=XC(T):DY=YC(T)
```

Add new missile.

```
600 IF STRIG(0)=0 AND M=0 THEN M=2:MX=
X:MY=Y:XM=2*DX:YM=2*DY
```

Move missile accordingly.

```
610 PLOT MX,MY:MX=MX+XM:MY=MY+YM:COLOR
 1:PLOT X,Y:LOCATE MX,MY,H:LOCATE MX-X
M/2,MY-YM/2,I:IF H+I>0 THEN 1000
620 COLOR M:PLOT MX,MY:IF X>75 THEN 30
00
630 GOTO GT
```

Make walls invisible or visible for higher missions.

```
640 IF V7*RND(1)<0.1 THEN SETCOLOR 2,0
,0
650 IF RND(1)<0.1 THEN SETCOLOR 2,9,4
```

Laser counting before firing.

```
690 IF T1>0 THEN T1=T1+1:IF T1>G*2 THE
N GOSUB 5000:T1=0
700 IF T2/2000+RND(1)<0.97 OR L=0 OR T
1>0 THEN 720
```

Set up laser for firing.

```
710 T1=1:COLOR 2:GOSUB 4000:SOUND 1,20
*G+20,2,4:TX=X:TY=Y:COLOR 0:GOSUB 4100
```

Regenerate a control box.

```
720 IF RND(1)<0.02 AND L=1 THEN H=INT(
10*RND(1)+1):COLOR 2:PLOT X(H),Y(H)
```

Regenerate laser.

```
900 IF L=0 THEN T4=T4+1:IF T4>G*30 THE
N L=1:COLOR 2:G=G-1:GOSUB 4000:COLOR 3
:GOSUB 4100:IF G<V8 THEN G=V8
910 T2=T2+1:T3=T3+1:IF T3>300 THEN T3=
0:G=G-1:IF G<V8 THEN G=V8
990 GOTO 500
```

Missile hits something.

```
1000 IF I>0 THEN MX=MX-XM/2:MY=MY-YM/2
1010 LOCATE MX,MY,H:ON H GOTO 2000,110
0,1150
```

Missile hits a control box.

```
1100 IF MY<7 AND MX>38 AND MX<42 THEN
1500
1110 V2=0:FOR T=40 TO 100 STEP 5:SOUND
 0,T,10,8:NEXT T:SOUND 0,0,0,0:SC=SC+1
:? U$;SC:GOTO 1200
```

Missile hits a wall.

```
1150 SOUND 0,100,10,4:FOR T=0 TO 20:NE
XT T:SOUND 0,0,0,0:V2=3
```

Get rid of missile.

```
1200 M=0:COLOR V2:PLOT MX,MY:XM=0:YM=0
:MX=0:MY=40:GOTO 630
```

Missile hits laser.

```
1500 IF V1=1 THEN V2=2:GOTO 1200
1510 V1=1:T1=0:L=0:T4=0:FOR T=0 TO 75:
SOUND 0,T,2,8:NEXT T:SOUND 0,0,0,0:SOU
ND 1,0,0,0
1520 SC=SC+5:? U$;SC:COLOR 0:GOSUB 400
0:GOSUB 4100:V2=0:GOTO 1200
```

Player destroyed.

```
2000 COLOR 2:PLOT X-2,Y-2:DRAWTO X+2,Y
+2:PLOT X-2,Y+2:DRAWTO X+2,Y-2:FOR T=1
 TO 40:SOUND 0,10,T,8:NEXT T
2010 COLOR 0:PLOT X-2,Y-2:DRAWTO X+2,Y
+2:PLOT X-2,Y+2:DRAWTO X+2,Y-2:SETCOLO
R 2,9,4:IF Y>37 OR Y<5 THEN Y=20
2020 D=D-1:? ,,,U$;D;:T1=0:X=5:SOUND 0
,0,0,0:SOUND 1,0,0,0:IF D<1 THEN POKE
53248,0:POKE 53249,0:GOTO 11000
2030 ? :GOSUB 6000:IF L=0 OR B=6 THEN
COLOR 0:GOSUB 4000
2040 GOSUB 4100:GOTO 500
```

Player reached right wall.

```
3000 IF B=5 THEN 8000
3010 IF B=6 THEN 10000
```

Count control boxes.

```
3020 I=0:FOR T=1 TO 10:LOCATE X(T),Y(T
),H:I=I+H:NEXT T:V4=V4+I/2
```

Transport player left if more than four control boxes.

```
3030 IF I>8 THEN SOUND 0,30,12,4:COLOR
 0:PLOT X,Y:X=5:FOR T=1 TO 100:NEXT T:
SOUND 0,0,0,0:GOTO 690
```

Give five point bonus for clearing all control boxes.

```
3040 IF I=0 THEN SC=SC+5:? U$;SC:FOR T
=1 TO 1200 STEP 20:SOUND 1,T,10,4:NEXT
 T
```

Reset board.

```
3050 FOR T=255 TO 0 STEP -1:SOUND 1,T,
10,5:NEXT T:X=5:T1=0:B=B+1:L=1:GOTO 20
0
```

Draw laser.

```
4000 PLOT 39,3:DRAWTO 41,3:PLOT 39,4:D
RAWTO 41,4:PLOT 40,5:RETURN
```

Draw laser's shield.

```
4100 PLOT 37,3:DRAWTO 37,7:DRAWTO 43,7
:DRAWTO 43,3:PLOT 36,3:DRAWTO 36,8:DRA
WTO 44,8:DRAWTO 44,3:RETURN
```

Fire laser.

```
5000 COLOR 2:PLOT 40,6:DRAWTO TX,TY:SO
UND 1,1,4,6:LOCATE X,Y,H
5020 COLOR 0:PLOT 40,6:DRAWTO TX,TY:IF
 H=2 THEN 2000
5030 COLOR 3:GOSUB 4100:SOUND 1,0,0,0:
RETURN
```

Draw perimeter.

```
6000 COLOR 3:FOR T=0 TO 3:PLOT T,T:DRA
WTO T,39+T:DRAWTO 79-T*0.6,39:DRAWTO 7
9-T*0.6,T:DRAWTO T,T:NEXT T:RETURN
```

Draw main computer room.

```
7000 GOSUB 6000:FOR T=10 TO 40 STEP 5:
PLOT T,0:DRAWTO T,40:NEXT T:FOR T=1 TO
 10:PLOT 7,T:DRAWTO 35,T:NEXT T
```

Move PM in view.

```
7010 POKE 53248,200:POKE 53249,200:FOR
 T=10 TO 40 STEP 5:PLOT 10,T:DRAWTO 40
,T:NEXT T:GOSUB 4100
7020 FOR T=43 TO 70 STEP 8:PLOT T,40:D
RAWTO T,8:PLOT T+4,0:DRAWTO T+4,35:NEX
T T
```

Draw 150 boxes.

```
7040 COLOR 2:FOR T=1 TO 150:PLOT 70*RN
D(1)+5,30*RND(1)+5:NEXT T:PLOT 43,30:G
OSUB 4000:GT=690:GOTO 500
```

Mission accomplished.

```
8000 SC=SC+30:? "MISSION ACCOMPLISHED"
:FOR T=255 TO 0 STEP -1:SOUND 0,T,10,6
:SOUND 1,T,12,6:NEXT T
```

Reset screen.

```
8010 GRAPHICS 5:POKE 752,1:SETCOLOR 0,
2,6:? "YOU NOW HAVE TO ESCAPE BEFORE":
? "THE COMPLEX IS DESTROYED."
```

Move PM out of view.

```
8020 POKE 53248,0:POKE 53249,0:FOR T=0
 TO 256:SOUND 0,T,10,4:NEXT T:? :? :?
SC,,,D:? U$;U$;
```

Draw walls.

```
8030 GOSUB 6000:FOR T=10 TO 60 STEP 20
:PLOT T,2:DRAWTO T,20+V4:PLOT T+10,40:
DRAWTO T+10,22-V4:V4=V4+2
8040 IF V4>16 THEN V4=16
```

Draw 100 boxes.

```
8050 NEXT T:X=5:B=6:GT=9000:V5=200-V9*
20:COLOR 2:FOR T=0 TO 100:H=8*RND(1):P
LOT H*H+5,30*RND(1)+5:NEXT T:V4=0:L=1
```

Print information labels.

```
8100 ? "SCORE","TIME LEFT","LIVES":?
```

Adjust escape time left.

```
9000 V5=V5-1:SOUND 1,V5,2,4:? ,U$;INT(
0.16*V5);"  ":IF V5>0 THEN 900
```

Didn't make it out.

```
9500 FOR T=10 TO 0 STEP -1:SETCOLOR 4,
0,12:FOR H=1 TO 10:NEXT H:SETCOLOR 4,0
,0:SOUND 0,T*10+100,12,10
9510 SOUND 1,T*5+125,2,10:NEXT T:SETCO
LOR 4,0,12:GOTO 11000
```

Escaped alive.

```
10000 SC=SC+10:X=5:GRAPHICS 2:? #6;"
 GOOD JOB!!"
10005 ? #6,,,,"  YOU COMPLETED","    M
ISSION ";V9,,,"SCORE",SC,"LIVES",D,"TI
ME LEFT ";V5*0.16
10010 FOR T=1 TO 80:SOUND 1,200*RND(1)
,10,4:FOR H=1 TO 20:NEXT H:NEXT T:SOUN
D 1,0,0,0
```

Adjust variables for next mission.

```
10020 V9=V9+1:B=V9:GT=640:T1=0:V8=V8-0
.3:V7=4.5-V9:GOTO 150
```

Start another game.

```
11000 SOUND 0,0,0,0:SOUND 1,0,0,0:? :?
 "      PRESS START TO PLAY AGAIN";
11010 IF PEEK(53279)<>6 THEN 11010
11020 RUN
```

ATARI® SWAT TABLE FOR:
SABOTAGE

LINES	SWAT CODE	LENGTH
100 - 210	EJ	608
220 - 540	WN	557
560 - 710	SX	598
720 - 1110	UA	573
1150 - 2000	WG	626
2010 - 3020	FQ	542
3030 - 4100	ZJ	610
5000 - 7010	RW	534
7020 - 8020	EC	606
8030 - 9500	GS	613
9510 - 11000	FA	576
11010 - 11020	KK	38

Ali-Baba And The Forty Thieves

A review by Guy S. Allred

By Stuart Smith (Quality Software). System requirements: 32K ATARI® 800 with disk. Suggested retail price: $32.95.

A burning, Arabian sun shines down on the white sands this fateful morning as a lowly royal messenger, Ali Baba, goes about his business, not knowing what adventures the day will bring. Soon a cry rings throughout the land: the Sultan Shahriar's daughter, Princess Buddir al-Buddoor, has been kidnapped by a ruthless band of thieves, led by the cunning Cogia Houssain. The Princess has been taken to the thieves' stronghold, deep inside a treacherous mountain. Can Ali Baba, aided by the Sultan's elite corps, rescue the Princess and return her to safety before nightfall?

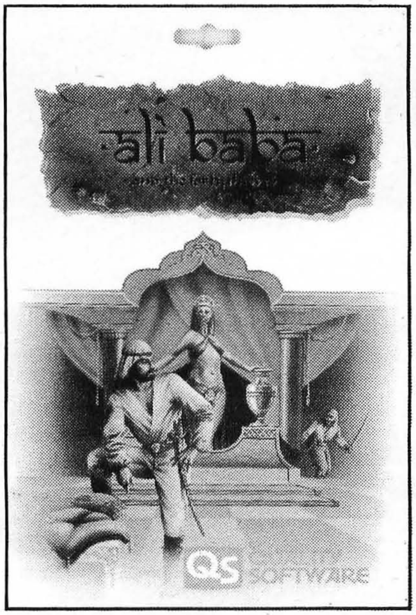
Such is the state of things when we meet Ali Baba in this new adventure/fantasy game from Quality Software. Stuart Smith has created a world filled with more characters than most novels, and enough escapade to interest almost anyone.

Little did I know what was in store for me on the morning my editor deposited Ali Baba on my desk. After initially spending over twelve hours playing the game, with no conscious notice of elapsed time, let me warn you, I am not about to write a bad review. This game is challenging, exciting, and well worth the price, not to mention the time you'll spend playing it.

The diskette comes with a 24 page documentation booklet designed to guide you through your options and to serve as a handy reference tool during play. Scenario, play option, characters (good or bad, friend or foe), and battle specifications are all fully outlined. You don't have to study, or even read, the booklet to play but it helps to familiarize yourself with the given information.

At the outset you are given the opportunity to select how you wish to play the game, either through the keyboard (arrow keys) or with joysticks. 1 to 17 people can play, each controlling as few as one character or as many as seventeen. I would recommend using joysticks for more than one player.

With each turn you are offered a variety of options. Resting lets you recoup energy lost from battle or injury. Adding characters provides a helpful diversion for you in making your way around enemies. Dropping gold or armor can free you from excess weight when in pursuit. These choices, to name but a few, are all given. The REST option is not offered at each turn. Obviously, when you are about to be pounced upon by an enemy you won't want to take a cat nap.



I found the ADD PLAYER option to be the most frequently used and the most helpful of all the options. It allows you to add as many characters as you please with each turn, or to retire those characters during their later turns. After you've spent several turns collecting gold, armor, and weapons, you may not want to risk death in battle. At this point you may want to add characters to create a necessary diversion, thus allowing you to slip by your enemies without fighting.

You can change the difficulty of the game at any time during play by adjusting the Monster Rebirth Rate. (Yes, they can come back to terrorize you at the least opportune of times.) Any level can be chosen, from 0 to 127, with the higher levels being more difficult. You're given quite a bit of freedom here and it's a definite asset. At times it comes as a comfort to know that once you've killed an enemy — he's dead!

The world you enter as Ali Baba is inhabited by a countless number of beings. The Sultan's elite corps, your allies, consists of 16 members ranging from dexterous humans to lithe elves, skilled halflings to powerful dwarves. A full description of each being is given in the documentation booklet and included in the character tables. This lets you see how each relates to all of the other creatures, both friend and foe.

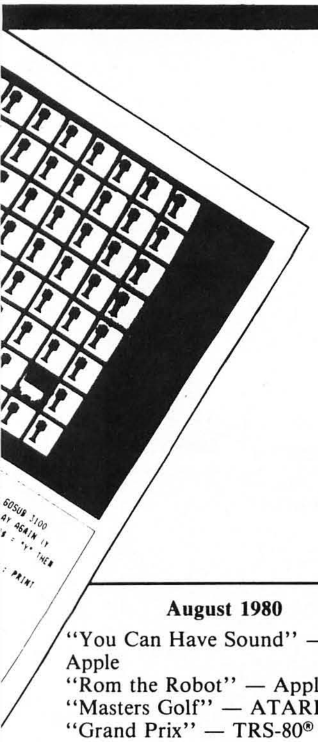
Sent to do battle, and to keep you from the princess, are, of course, the theives and the various creatures who dwell in the mountain stronghold. These creatures include felines and other mortal animals, as well as magical, metaphysical beings. Not all of the mountain dwellers are adverse to the forces of good and many can offer needed aid in times of trouble. All of these creatures are also included in the character tables.

The graphics are both nicely executed and refreshingly different. As you enter each new room, during the early portion of the game, the contents, including visible exits, are identified for you using a bright "halo" and titles. This helps you to become familiar with the various

continued on page 82

APPLE CAPTURE by William J. Ryan
TRS-80™ Version
SoftSide
Your BASIC Software Magazine • NUMBER ELEVEN
SoftSide
January 1982 Vol. 5, No. 4 THREE DOLLARS
Off and running with GAMBLER
SoftSide
TITAN
SoftSide
THREE DOLLARS
Special Issue: Music in the Micro
Flight of the Bumblebee

Back Issues from SoftSide™



VOLUMES FROM THE PAST

If you like what this issue of **SoftSide** has to offer, you should see what's waiting for you in **SoftSide** back issues! You may feel that you've missed out on many of our programs that appeared before you became a subscriber. It's not too late to do something about it. You won't have to miss a thing because we still have back issues available to complete your **SoftSide** library! But, order now as some of our more popular issues are already out of stock and others are dwindling quickly.

Listed below are all of our past issues with their FEATURE programs and the systems they're for. For a more complete index of all the programs and articles offered in each of the back issues of **SoftSide** please refer to the May, 1981, issue. Each issue costs $3.50 for the magazine only. These issues are also available with the programs on cassette for $9.95 or on disk for $14.95 (except DV issues).

The enhanced Disk Versions (DV) contain an extra program for each system. The TRS-80® DV began with the September, 1981, issue. The Apple DV began October, 1981, and the ATARI® DV started in November, 1981. Each enhanced DV costs $19.95.

August 1980

"You Can Have Sound" — Apple
"Rom the Robot" — Apple
"Masters Golf" — ATARI®
"Grand Prix" — TRS-80®

October 1980

"Developing Data Base II"
All Systems
"Moonlanding" — Apple
"World Series" — ATARI®
"Earth-Port II" — TRS-80®

November 1980

"Developing Data Base III" — All Systems
"Collision" — Apple
"Trench" — ATARI®
"Kriegspiel" — TRS-80®

December 1980

"Developing Data Base IV"
All Systems
"Baseball" — Apple
"Speedello" — ATARI®
"Kidnapped" — TRS-80®

January 1981

"Developing Data Base V"
All Systems
"Convoy" — Apple and TRS-80®
"Angle Cannon" — ATARI®
"Ship Destroyer" — TRS-80®

February 1981

"Developing Data Base VI"–
All Systems
"Miner" — All Systems
"Mini-Golf" — ATARI® and TRS-80®
"Long Distance" — TRS-80®

March 1981

"Developing Data Base VII" — All Systems
"Strategy Strike" — Apple and TRS-80®
"Flags" — ATARI®
"Volcano" — TRS-80®

April 1981

"Battle At Sea" — Apple
"Convoy" — ATARI®
"Dominoes" — TRS-80®

May 1981

"Galaxia" — Apple
"Dodge" — ATARI®
"Orienteering At Jacque's Coulee" — TRS-80®

June 1981

"Old Glory" — All Systems
"Word-Search Puzzle Generator" — All Systems
"Anallist" — TRS-80®

October 1981

"Leyte" — All Systems
"Developing Data Base"—Apple
"Character Generator" — ATARI®
"Envyrn™" — TRS-80®

Enhanced Disk Versions

"Super Dairy Farming" — Apple
"Gameplay" — TRS-80®

November 1981

"Flight of the Bumblebee" — All Systems
"Music Machine" — Apple
"Music Programmer"—ATARI®
"Music Editor" — TRS-80®

Enhanced Disk Versions

"National Anthems" — Apple
"Volleyball" — ATARI®
"Mean Checkers Machine" — TRS-80®

December 1981

"Titan" — All Systems
"Aircraft Commander" — Apple
"Developing Data Base" — ATARI®
"Electronics Assistant" — TRS-80®

Enhanced Disk Versions

"Bobsledding" — Apple
"Survive" — ATARI®
"Konane" — TRS-80®

January 1982

"Gambler" — All Systems
"Microtext 1.1" — All Systems
"Apple Capture" — Apple
"Piazza Hotel" — ATARI®
"TRS-Man" — TRS-80®

Enhanced Disk Versions

"Nuclear Submarine Adventure"– Apple, TRS-80®
"Death Star" — ATARI®

February 1982

"Space Rescue" — All Systems
"Rubicube" — Apple
"Defense" — ATARI®
"Maze Sweep" — TRS-80®

Enhanced Disk Versions

"Andorra" — Apple
"Kismet II" — ATARI®
"Help Package" — TRS-80®

March 1982

"Hexapawn" — All Systems
"Magical Shape Machine" — Apple
"Outer Space Attack" — ATARI®
"Killer Cars" — TRS-80®

Enhanced Disk Version

"PEEKER/POKER" — Apple
"Curse of the Pharaoh" — ATARI®
"Warpath" — TRS-80®

April 1982

"Microtext" — All Systems
"Poster Maker" — Apple
"ATARI® Banner Machine" — ATARI®
"Database" — TRS-80®

Enhanced Disk Versions

"Semaphore" — Apple
"Renumbering for the ATARI®" — ATARI®
"Screen Print" — TRS-80®

May 1982

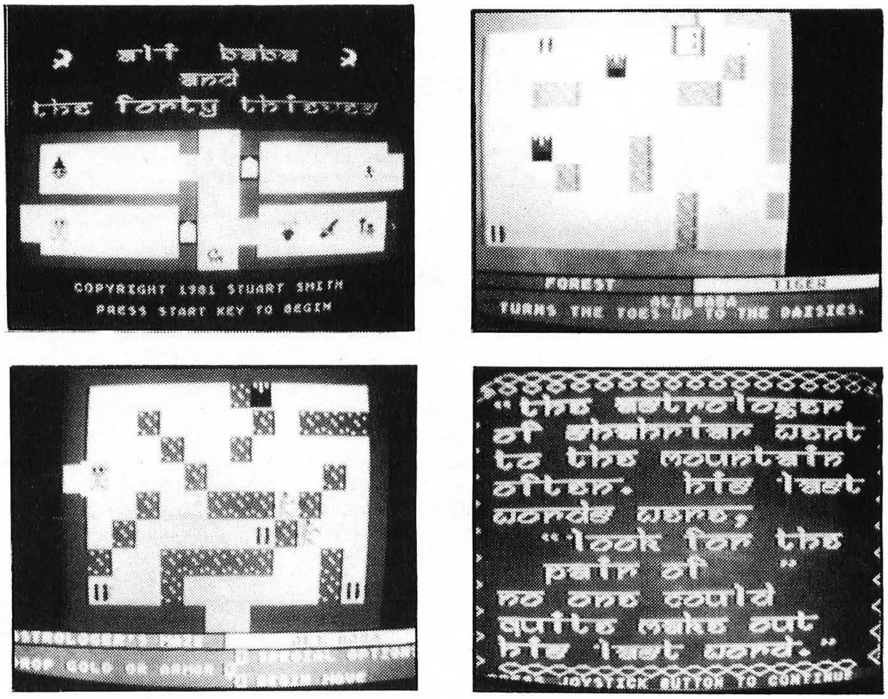
"Solitaire" — All Systems
"Micro-Man" — Apple
"Cross Reference" — ATARI®
"Ladders" — TRS-80®

Enhanced Disk Versions

"List Formatter" — Apple
"Robot Battle" — ATARI®
"Breakthru" — TRS-80®

Use card provided in this issue to order or send a list of the back issues you'd like, with payment of $3.50 per magazine (minimum order of 3 issues) to: **SoftSide Publications, 515 Abbott Drive, Broomall, PA 19008**

To order the magazine/media combinations, use the card provided or send $9.95 per cassette and magazine, $14.95 per disk and magazine, or $19.95 per DV and magazine to: **SoftSide Publications, 6 South Street, Milford, NH 03055**
Prices for USA orders only. For foreign orders, see page 16.



ali baba
and
the forty thieves

COPYRIGHT 1981 STUART SMITH
PRESS START KEY TO BEGIN

ALI BABA
TURNS THE TOES UP TO THE DAISIES.

"the astrologer of shahrian went to the mountain often. his last words were, "look for the palm of " no one could quite make out his last word."

JOYSTICK BUTTON TO CONTINUE

Screen shots from ***Ali Baba and the Forty Thieves***

symbols used. You are always told where you are and, through change of color, made aware of the changes in both level and danger as you delve deeper into the mountain.

Scattered throughout the game are symbols indicating runes. Landing on these symbols will present you with a graphics display, accompanied by music, offering information useful to you in your quest. The style of lettering used in these displays is reminiscent of ancient Arabic writings. The accompanying music also helps to enhance the Middle Eastern flavor, by conjuring pictures of street markets and fakirs. It's this kind of attention to detail, used throughout the game, which has made this an excellent program.

If there is one drawback to the game, it's the speed with which it executes. If you enter a room filled with several opponents and characters added to fight them, you must sit through each of their turns, foe as well as friend. This adds quite a bit of playing time, but without this option, the game would not be nearly so distinctive.

If all of this sounds just a little familiar to fans of the *DUNGEONS & DRAGONS*™ games, it should. The game bears more than a passing resemblance to the *DUNGEONS & DRAGONS*™ games. The distinct difference is that you share the Dungeon Master's functions with the computer, as well as serving as a player. The computer decides all battles while the player retains control over most of the rest of the game.

Like the *DUNGEONS & DRAGONS*™ games, *Ali Baba* is of considerable length, but with the convenient option of SAVE TO DISC, you can stop the game at any time and return to it when you please, exactly where you left it.

In speaking to Bob Christiansen at Quality Software, I learned that the Apple version is in the editing stage and should be out as this magazine goes to press. This version will use paddles rather than joysticks.

At this point, I shall leave you to enjoy the rest of this issue while I return to Arabia, Ali Baba, and whatever else awaits me. I recommend you join me as soon as you can buy a copy of *ALI BABA AND THE FORTY THIEVES*. I'll be waiting.

"DUNGEONS & DRAGONS™ is a trademark owned by and used under license from TSR Hobbies, Inc."

K-Byter

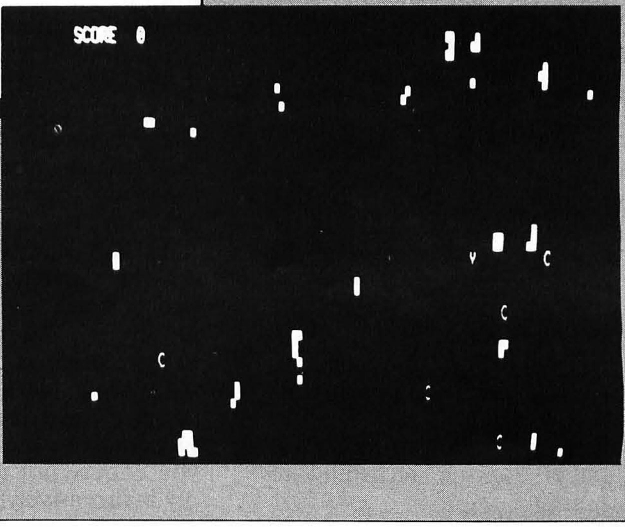
Meteor Dodge

A TRS-80® K-Byter by Matt Hillman, Brookline, MA.

The object of *Meteor Dodge* is to pick up a specified number of fuel canisters before you are destroyed by a meteor storm. You choose the number of canisters at the beginning of the game, suiting the difficulty of the game to your mood and experience.

You are represented on the screen by the letter Y, and the canisters by the letter C. The other characters which appear on the screen are the meteors which you must avoid. You pick up canisters by moving over them with your Y, which you move using the arrow keys.

```
5 CLS
10 PRINT @25,"METEOR DODGE":YV=8:YH=32:INPUT "HOW MANY CANISTERS
 DO YOU WANT TO TRY TO RESCUE";C:PRINT "YOU ARE THE Y AND THE C'
S ARE CANISTERS.  ANYTHING ELSE IS A    METEOR. ------------HIT
ANY KEY TO START-----------------------"
20 A$=INKEY$:IFA$=""THEN20
30 CLS
35 FORI=1TOC:X=RND(1023)-1:IFPEEK(15360+X)=32THENPRINT@X,"C";:NE
XTIELSEI=I-1:NEXTI
37 PRINT@YV*64+YH,"Y";
40 FORI=1TO2
45 PRINT@0,"SCORE ";S;
50 A=PEEK(14400):IFA=16ORA=8ORA=32ORA=64THEN99
60 D=RND(1023)-1:IFPEEK(15360+D)=32THENPRINT@D,CHR$(RND(63)+128)
;ELSEGOTO60
70 GOTO40
99 PRINT@YH+YV*64," ";
100 IFA=8THENYV=YV-1ELSEIFA=16THENYV=YV+1ELSEIFA=32THENYH=YH-1EL
SEYH=YH+1
120 IFYV<0ORYV>15ORYH<0ORYH>63THENCLS:PRINT@20,"YOU ARE FOREVER
LOST IN SPACE":END
130 IFPEEK(15360+(64*YV)+YH)>128THENCLS:PRINT@25,"YOU HIT A METE
OR":ENDELSEIFPEEK(15360+(64*YV)+YH)<>32THENS=S+1:IFS=CTHENCLS:PR
INT@28,"YOU WIN!":END
140 PRINT@64*YV+YH,"Y";:NEXTI:GOTO60
```

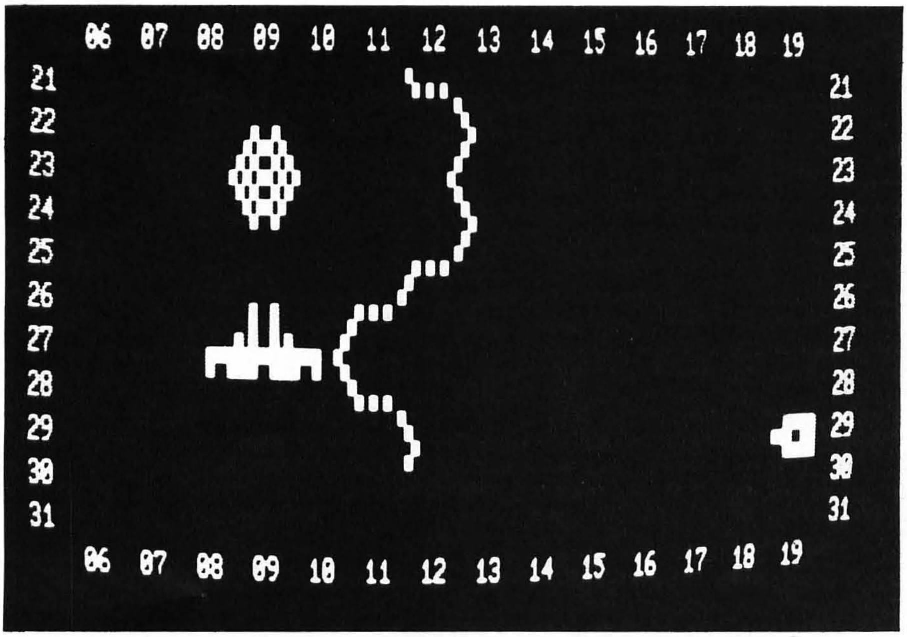


SCORE 0

SoftSide

TRS-80® DV

KRIEGSPIEL II



by Ron Potkin

***Kriegspiel II* is a graphics wargame simulation for a TRS-80® Model I or III with at least 32K RAM and disk drive. It is included as a bonus program on this month's TRS-80® Disk Version.**

The objective of this wargame is to enter your opponent's city or to reduce the number of opposing pieces to less than half of your own. It is played on a map which is just under four times the size of the video screen, consisting of 31 rows of 16 hexagons. The area seen on the screen at any one time is 11 rows of 7 hexagons, or one-sixth of the map.

Getting Started

The map has the following features:

(1) Two opposing cities: one in the northwest and one in the southeast.

(2) Five neutral towns (in the same locations in all scenarios).

(3) A river flowing from north to south (in varying locations in different scenarios).

(4) Mountains in various locations, the number varying according to your choice.

Each army contains 20 pieces:

Piece	Number	Movement Factor	Attack Factor
Heavy Tank	4	14	3
Light Tank	6	12	2
Infantry	10	10	1

This number may be increased up to a limit of 25 through recruiting, which will be explained below.

Upon running the program you will be prompted to enter the scenario number. You may enter any number up to 9999, which will then be used as the basis for creating the map features and weather. Re-using the same number will allow you to re-create particularly interesting scenarios. You are then asked to choose a number of mountains; any number up to about 200 may be used. The screen then clears and you will see the various pieces: mountains, cities, towns, and river, being placed on the map. After this is finished, you are given the option of reviewing the map; obviously the game will be more of a challenge if you choose not to do so. If you reply affirmatively, the map will scroll across the screen until you press a key.

The date, weather, and details of the scenario are displayed at the top of the screen; the number of pieces on each side is displayed at the bottom. The southeast corner of the map is displayed (Southeast always moves first), and play begins.

The order of play is as follows:

(1) Movement phase: The Southeast player moves all, some, or no pieces.

(2) Combat phase: All conflicts resulting from the Movement phase are resolved.

(3) The Movement and Combat phases are repeated for the Northwest player.

(4) The player holding a majority of towns recruits an additional piece.

(5) The calendar is updated and the weather for the next month is displayed.

These five steps are repeated throughout the game.

The Movement Phase

A player may, during his or her turn, move pieces up to the limit of the movement factor given in the chart above. It takes two movement units to move from one hexagon to the adjacent one, with some exceptions. The six hexagons surrounding a city or town are considered major roads, and it takes only one movement unit to move OUT of one of these. You cannot move into a town hexagon or into your own city. You cannot move into a mountain hexagon, nor even next to one if there are floods. Crossing the river, except at a bridge, normally takes four units. If there is ice, it takes only two, and if there are floods the river cannot be crossed at all. Note, however, the effect of a city or town lying across a river (scenario 8). These factors may also be affected by the weather, as described below.

Six number keys and seven letter keys are used to effect movement and various actions during this phase. To move a piece, type in a direction number from 1 through 6.

You can picture these numbers as the points on a six-hour clock. Northeast is 1, southeast is 2, south is 3, southwest is 4, northwest is 5, and north is 6. A piece cannot be moved further when it moves next to an opposing piece, when its movement factor reaches zero, or when its movement factor reaches one and it is not in a hexagon adjacent to a town or city. (But note option W below.)

Here are the available options during the Movement phase. They are selected by pressing the indicated key:

(S) Stop a piece before it has expended all movement units.

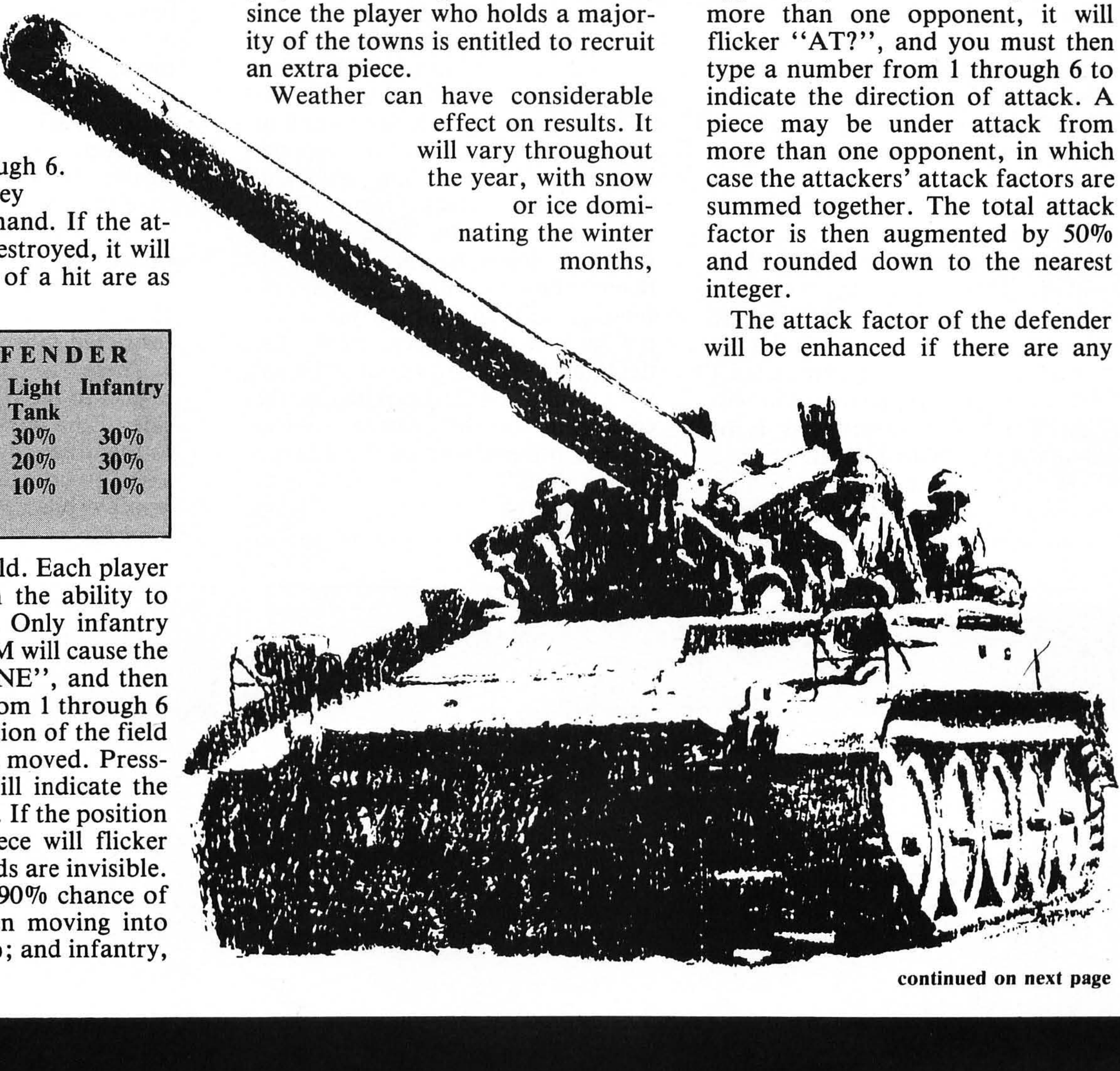
(C) Change sides, if you decide that you have finished all the moves necessary even though some pieces have not yet moved.

(H) Hold further movement of a piece temporarily so that pieces can be moved other than in their normal, pre-defined order. This makes it easier, for example, to clear a congested area.

(F) Fire at another piece two hexagons away. This may be done once during a piece's movement. After pressing F, press a direction number from 1 through 6. Pressing any other key will cancel the command. If the attacked piece is not destroyed, it will fire back. The odds of a hit are as follows:

	DEFENDER		
Attacker	Heavy Tank	Light Tank	Infantry
Heavy Tank	20%	30%	30%
Light Tank	10%	20%	30%
Infantry	10%	10%	10%

(M) Lay a mine field. Each player starts the game with the ability to lay five mine fields. Only infantry may do so. Pressing M will cause the piece to flicker "MINE", and then pressing a number from 1 through 6 will specify the direction of the field from the piece being moved. Pressing any other key will indicate the number of mines left. If the position is not vacant the piece will flicker "HOW?". Mine fields are invisible. Heavy tanks have a 90% chance of being destroyed when moving into one; light tanks, 50%; and infantry, 30%.



(T) Transport an infantry piece from one of the six hexagons surrounding one town or city, to the vicinity of another town or city. The destination city/town must be friendly and must have an empty hexagon adjacent to it. This takes four movement points and cannot be done during ice or snow. The destination is chosen by typing a number from 1 to 7, corresponding to these locations:

```
   1        2   3
          4
    5   6        7
```

(W) Wait. There may be times when it is desirable to move the full distance and then fire or lay a mine field. "Wait" avoids automatic continuation, and pauses until you press either (S)top or (C)ontinue to attack.

A town is considered occupied if the surrounding hexagons contain forces from both sides. If only one side resides on these positions, it is considered held or friendly. It is to a player's advantage to hold towns, since the player who holds a majority of the towns is entitled to recruit an extra piece.

Weather can have considerable effect on results. It will vary throughout the year, with snow or ice dominating the winter months, and floods playing a significant role in the spring. There is a chance of rain throughout other months. Here are the effects of the weather conditions:

Fine: Normal conditions.

Rain: Movement factor is halved.

Snow: Movement factor is halved; no transport is possible.

Ice: No transport is possible; river is normal terrain.

Floods: River can be crossed only at bridges; no movement is possible next to mountains.

Combat Phase

Combat is divided into a number of rounds. It takes place upon completion of the Movement phase, between opposing pieces on adjacent hexagons, and will continue until there are no more adjacent opponents. Most of this phase is automatic and needs no intervention, unless a decision is needed between options (e.g., attack, advance, retreat, or eliminate).

Each piece will attack only one opposing piece. If it is adjacent to more than one opponent, it will flicker "AT?", and you must then type a number from 1 through 6 to indicate the direction of attack. A piece may be under attack from more than one opponent, in which case the attackers' attack factors are summed together. The total attack factor is then augmented by 50% and rounded down to the nearest integer.

The attack factor of the defender will be enhanced if there are any

continued on next page

friendly pieces adjacent to the opposing pieces which are attacking. If such friendly pieces are under attack themselves, however, they will be of no support value. If a particular piece is in a supportive position for more than one attack, it will be of actual benefit in only one. The defender's attack factor will also be enhanced (doubled, in fact) if the attack is coming exclusively from across the river.

After calculating the attacker's and defender's attack factors in this way, their ratio is reduced to a simple form and the table below is consulted.

The numbers in the table indicate five possible outcomes of an attack. Given a particular ratio of attack forces, one of the letters in the appropriate column is chosen at random. (Odds greater than 6:1 against a piece mean automatic elimination.) The possible outcomes are:

(1) All attackers are eliminated.

(2) All attackers must retreat two hexagons. In retreating they must move to a vacant location which is not adjacent to an opposing piece. They may cross the river. Both flood and mine field conditions are in operation during retreats (and advances). If a piece is unable to retreat, press E and the piece will be eliminated.

(3) Exchange. The defender is eliminated. The attacker must choose one or more of his own pieces to eliminate, such that their total attack factor is at least equal to that of the eliminated defender. (If this is not possible, then all the attacking pieces will be eliminated.) Each candidate for elimination will flicker "EX" in turn; press E to eliminate it or N to keep it.

(4) Defender must retreat two hexagons. The same conditions apply as for the attacker in (2). The attacking forces may then advance one hexagon in any direction, including moving next to an opposing piece to attack in the next round. If you do not want a piece to advance, press S.

(5) Defender is eliminated and the attacker may advance one hexagon. Note that supporting defenders are not involved in the outcome, and are not eliminated or forced to retreat.

At the end of each round of play, the occupation status of each town is checked. The side which holds a majority of towns will recruit an extra piece: infantry for a majority of one town, a light tank for a majority of two, and a heavy tank for a majority of three or more. Recruiting will not occur if the player with the majority already has 25 pieces, or if all the hexagons surrounding his city are occupied.

Strategy Notes

Think before you fire. A light tank firing on a heavy tank, for example, has a 10% chance of hitting; but the heavy tank has a 20% chance when it fires back.

Calculate odds carefully prior to combat. Rounding down takes place twice. For example, consider two heavy tanks and an infantry unit attacking a heavy tank with a supporting infantry unit. The attacking units' total attack factor is 7 (3+3+1). With the 50% addition, rounded down to the nearest integer, this becomes 10. The defending tank and supporting infantry have a total factor of 4. The defender:attacker ratio of 4:10 will then become 1:2. Looking at this column in the table, you can see that there is a possibility of the attacker being completely eliminated (outcome number 1). Adding a single additional infantry unit to the attacking force, however, would yield an attack factor of 12 (3+3+1+1, plus 50%), giving a much more favorable final ratio of 4:12 or 1:3, with no possibility of total elimination of the attacker.

Guard your city carefully. Infantry is not always enough. Attack by a heavy tank can create havoc, since it can continually attack and advance during the combat phase until it enters the city (although it may not advance into the city in that turn).

Only infantry can lay or destroy mine fields effectively, so they must be conserved and not sent on desperate missions.

The hexagonal "honeycomb" playing board may be confusing until you get used to it. Pieces which line up horizontally with one another may appear to be adjacent, but actually are not. Thus a piece may lie to the immediate east or west of a town, but not be occupying one of its adjacent hexagons.

The number of scenarios is almost unlimited. Although the program will place mountains only on vacant hexagons, it may place them in such a way that they surround towns or pieces, or block bridges. The lack of such restrictions makes the program more varied and interesting; however, if you don't like a particular scenario, it takes very little time to create a new one.

Try a scenario with 200 mountains. This creates some interesting features: passes, valleys, an occasional open plain, and very restricted movement.

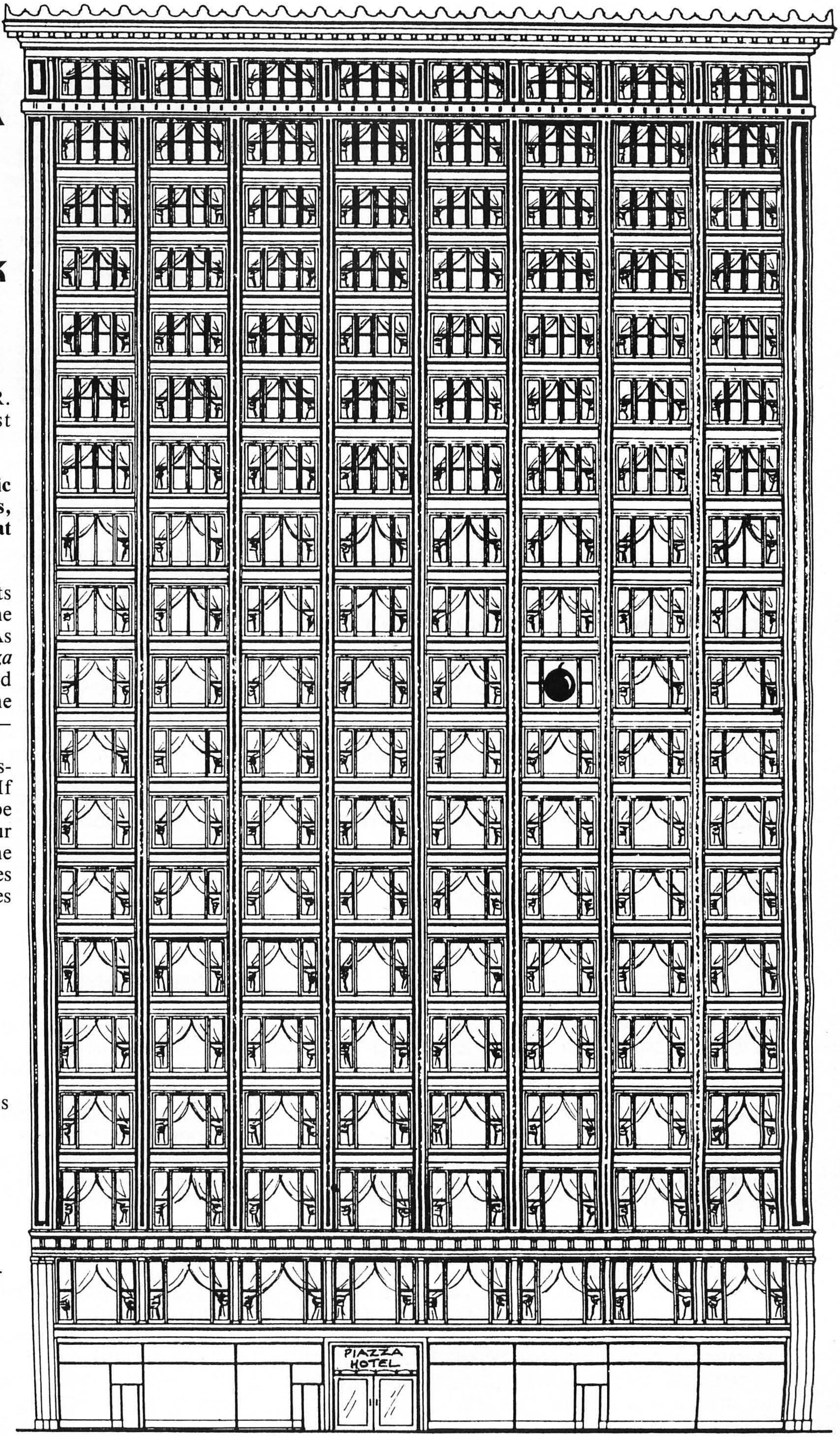
There is one limitation with regard to the river. If it runs off the side of the map, it will wrap around to the other edge. If this happens, it loses its orientation and the map is unplayable. This very seldom happens.

ATTACK FACTOR RATIO --DEFENDER:ATTACKER

6:1	5:1	4:1	3:1	2:1	1:1	1:2	1:3	1:4	1:5	1:6
---	---	---	---	---	---	---	---	---	---	---
1	1	1	1	1	1	1	3	3	4	4
1	1	1	1	1	1	2	3	4	4	5
1	1	1	2	2	2	3	4	4	5	5
1	1	2	2	2	3	3	4	5	5	5
1	2	2	2	3	4	4	5	5	5	5
2	2	2	2	3	5	5	5	5	5	5

PIAZZA HOTEL

By Gary Dominick



TRS-80® version By Michael R. Freifeld, Translation Contest Winner.

***Piazza Hotel* is a game of logic and luck, with sound and graphics, for a TRS-80® Model I or III with at least 16K RAM.**

This program was published in its original ATARI® version in the January, 1982, issue of ***SoftSide***. As the house detective of the *Piazza Hotel*, you have just been informed that there is a bomb in one of the rooms. Your job is to find it — FAST.

You begin your search by choosing a floor and a room number. If you are not correct, you will be given clues to help you focus your search. You never know what the next guess will lead to. Sometimes it's the correct room. Sometimes it's...BOOM!

Variables

A: String variable used for "Piazza Hotel" banner and the other major graphics characters.
B: String variable used for player's rating and other purposes.
C, D: Miscellaneous string variables.
E: Used in sound routine and in generation of graphics characters.
F-Z: Used initially to generate graphics characters.
F, R: The floor and room number of the bomb.
G, H: The player's guess for the floor and room number.
N: The number of guesses left.
Q: Value of the total number of guesses.
W-Z: Used in FOR-NEXT loops.

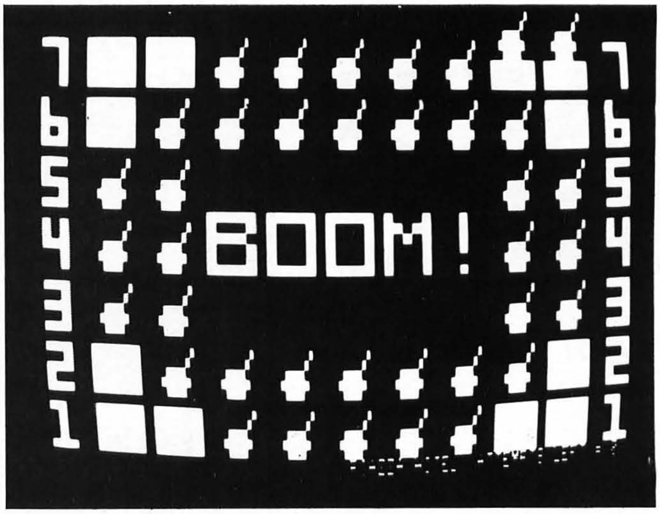
```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         TRS-80 BASIC         $
$        "PIAZZA HOTEL"        $
$   AUTHOR: GARY DOMINICK      $
$ TRANS: MICHAEL R. FREIFELD $
$    (C) 1982    SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Define variables. Set up Piazza Hotel banner.

```
10 CLEAR4000:DEFSTRA-D:DEFINTF-Z:I=128:J=131:L=176:N=191:P=135:Q
=186:R=149:S=179:U=171:V=151:W=8:Y=26:F=170:G=184:H=181:E=177:T=
163
20 C=CHR$(W):FORX=1TO52:C=C+CHR$(W):NEXTX
30 B=CHR$(Y)+C
40 D=CHR$(N)+CHR$(J)+CHR$(J)+CHR$(N)+CHR$(I)+CHR$(J)+CHR$(U)+CHR
$(V)+CHR$(J)+CHR$(I)+CHR$(N)+CHR$(J)+CHR$(J)+CHR$(N)+CHR$(I)
50 FORX=1TO2:D=D+CHR$(J)+CHR$(T)+CHR$(N)+CHR$(P)+CHR$(I):NEXTX
60 D=D+CHR$(N)+CHR$(J)+CHR$(J)+CHR$(N)+CHR$(I)+CHR$(I)+CHR$(I)+C
HR$(N)+CHR$(I)+CHR$(I)+CHR$(N)+CHR$(I)+CHR$(N)+CHR$(J)+CHR$(J)+C
HR$(N)+CHR$(I)
70 D=D+CHR$(J)+CHR$(U)+CHR$(V)+CHR$(J)+CHR$(I)+CHR$(N)+CHR$(J)+C
HR$(J)+CHR$(J)+CHR$(I)+CHR$(N)+B
80 D=D+CHR$(N)+CHR$(J)+CHR$(J)+CHR$(J)+CHR$(I)+CHR$(L)+CHR$(Q)+C
HR$(H)+CHR$(L)+CHR$(I)+CHR$(N)+CHR$(J)+CHR$(J)+CHR$(N)+CHR$(I)
90 FORX=1TO2:D=D+CHR$(G)+CHR$(N)+CHR$(E)+CHR$(L)+CHR$(I):NEXTX
100 D=D+CHR$(N)+CHR$(J)+CHR$(J)+CHR$(N)+CHR$(I)+CHR$(I)+CHR$(I)+
CHR$(N)+CHR$(J)+CHR$(J)+CHR$(N)+CHR$(I)+CHR$(N)+CHR$(L)+CHR$(L)+
CHR$(N)+CHR$(I)
110 D=D+CHR$(I)+CHR$(F)+CHR$(R)+CHR$(I)+CHR$(I)+CHR$(N)+CHR$(S)+
CHR$(S)+CHR$(L)+CHR$(I)+CHR$(N)+CHR$(L)+CHR$(L)+CHR$(L)
120 CLS:PRINT@4,D;:GOSUB1130
```

Initialize sound routine (by Alan J. Zett, SoftSide, November, 1981.)

```
140 Z=0:FORX=1TO158:READY:Z=Z+Y:NEXT:IFZ<>15204THENCLS:PRINT"DAT
A ERROR":STOP:ELSEY=86:X=255:POKE-1,0:IFPEEK(-1)<>0THENX=191:POK
E-16385,0:IFPEEK(-16385)<>0THENX=127
150 POKE16562,X:POKE16561,Y:CLEAR5000:W1=PEEK(16561)+2:W2=PEEK(1
6562):W=W1+W2*256:Z=W-1:FORX=1TO158:Z=Z+1:Z=Z+65536*(Z>32767)
160 READY:IFY<0THENY=W1+ABS(Y):POKEZ,Y+256*(Y>255):Z=Z+1:POKEZ,W
2-(Y>255):NEXTELSEPOKEZ,Y:NEXT
170 IFPEEK(16396)=201:POKE16526,W1:POKE16527,W2:ELSECMD"T":DEFUS
R=W1+(W2+256*(W2>127))*256:POKE14308,0
180 IFPEEK(16807)+PEEK(16808)*256<>W+24THENW=USR(0)
190 SOUND11,11
200 DATA58,166,65,50,-164,42,167,65,34,-165,62,195,50
210 DATA166,65,33,-24,34,167,65,201,245,123,254,2,40,4,254
220 DATA16,32,79,229,213,42,230,64,126,183,32,4,35,35,35,35
230 DATA215,6,5,17,-156,26,190,32,104,19,35,16,248,43,215
240 DATA43,34,230,64,241,241,241,241,197,213,215,205,55,35
250 DATA229,205,127,10,42,33,65,34,-167,225,215,43,34,230,64
260 DATA35,205,55,35,43,229,205,127,10,42,33,65,58,-167,60
270 DATA183,87,24,4,24,48,24,44,66,62,1,211,255,16,252,66,62
280 DATA2,211,255,16,252,58,64,56,230,4,32,7,124,181,40,3,43
290 DATA24,228,175,50,154,64,225,209,193,215,195,30,29,83,79
300 DATA85,78,68,209,225,241
```



Initialize variables and set up graphics characters for use in program.

```
310 DEFSTRA-D:DEFINTF-Z:DIMA(22),B(6):F=178:G=156:H=130:I=128:J=
131:K=140:L=176:M=188:N=191:O=136:P=160:Q=132:R=174:S=179:T=157:
U=167:V=147:W=8:Y=26:Z=143:B=CHR$(Y)+CHR$(W)+CHR$(W)+CHR$(W)
```

Graphics characters 1-7.

```
320 A(1)=CHR$(K)+CHR$(M)+CHR$(I)+B+CHR$(L)+CHR$(N)+CHR$(L)
330 A(2)=CHR$(K)+CHR$(K)+CHR$(M)+B+CHR$(N)+CHR$(S)+CHR$(S)
340 A(3)=CHR$(K)+CHR$(K)+CHR$(M)+B+CHR$(S)+CHR$(S)+CHR$(N)
350 A(4)=CHR$(M)+CHR$(I)+CHR$(M)+B+CHR$(J)+CHR$(J)+CHR$(N)
360 A(5)=CHR$(M)+CHR$(K)+CHR$(K)+B+CHR$(S)+CHR$(S)+CHR$(N)
370 A(6)=CHR$(M)+CHR$(I)+CHR$(I)+B+CHR$(N)+CHR$(S)+CHR$(N)
380 A(7)=CHR$(K)+CHR$(K)+CHR$(M)+B+CHR$(I)+CHR$(I)+CHR$(N)
```

Graphics symbols and arrows.

```
390 B=B+CHR$(W)+CHR$(W)
400 A(8)=CHR$(I)+CHR$(I)+CHR$(I)+CHR$(P)+CHR$(Q)+B+CHR$(I)+CHR$(
R)+CHR$(N)+CHR$(T)+CHR$(I)
410 A(9)=CHR$(I)+CHR$(O)+CHR$(G)+CHR$(I)+CHR$(I)+B+CHR$(H)+CHR$(
S)+CHR$(U)+CHR$(V)+CHR$(I)
420 A(10)=CHR$(M)+CHR$(M)+CHR$(M)+CHR$(M)+CHR$(M)+B+CHR$(N)+CHR$
(N)+CHR$(N)+CHR$(N)+CHR$(N)
430 A(11)=CHR$(I)+CHR$(I)+CHR$(K)+CHR$(L)+CHR$(I)+B+CHR$(J)+CHR$
(J)+CHR$(S)+CHR$(Z)+CHR$(J)
440 A(12)=CHR$(I)+CHR$(L)+CHR$(K)+CHR$(I)+CHR$(I)+B+CHR$(J)+CHR$
(Z)+CHR$(S)+CHR$(J)+CHR$(J)
450 A(13)=CHR$(I)+CHR$(I)+CHR$(M)+CHR$(I)+CHR$(I)+B+CHR$(J)+CHR$
(K)+CHR$(N)+CHR$(K)+CHR$(J)
460 A(14)=CHR$(I)+CHR$(L)+CHR$(M)+CHR$(L)+CHR$(I)+B+CHR$(J)+CHR$
(I)+CHR$(N)+CHR$(I)+CHR$(J)
470 A(15)=CHR$(I)+CHR$(K)+CHR$(K)+CHR$(M)+CHR$(M)+B+CHR$(L)+CHR$
(K)+CHR$(J)+CHR$(I)+CHR$(Z)
480 A(16)=CHR$(M)+CHR$(M)+CHR$(K)+CHR$(K)+CHR$(I)+B+CHR$(Z)+CHR$
(I)+CHR$(J)+CHR$(K)+CHR$(L)
490 A(17)=CHR$(K)+CHR$(L)+CHR$(I)+CHR$(I)+CHR$(L)+B+CHR$(I)+CHR$
(L)+CHR$(S)+CHR$(M)+CHR$(N)
500 A(18)=CHR$(L)+CHR$(I)+CHR$(I)+CHR$(L)+CHR$(K)+B+CHR$(N)+CHR$
(M)+CHR$(S)+CHR$(L)+CHR$(I)
```

```
510 A(19)=CHR$(N)+CHR$(S)+CHR$(S)+CHR$(S)+CHR$(Z)+B+CHR$(N)+CHR$
(L)+CHR$(L)+CHR$(L)+CHR$(N)
520 A(20)=CHR$(N)+CHR$(J)+CHR$(J)+CHR$(J)+CHR$(N)+B+CHR$(N)+CHR$
(L)+CHR$(L)+CHR$(L)+CHR$(N)
530 A(21)=CHR$(N)+CHR$(K)+CHR$(L)+CHR$(K)+CHR$(N)+B+CHR$(N)+CHR$
(I)+CHR$(I)+CHR$(I)+CHR$(N)
540 A(22)=CHR$(I)+CHR$(I)+CHR$(N)+CHR$(I)+CHR$(I)+B+CHR$(I)+CHR$
(I)+CHR$(S)+CHR$(I)+CHR$(I)
```

Initialize variables for player rating.

```
550 B(1)="    SUPER SLEUTH":B(2)=" MASTER DETECTIVE":B(3)="DETEC
TIVE CLASS-A":B(4)="DETECTIVE CLASS-B":B(5)="DETECTIVE CLASS-C":
B(6)="   NOVICE TRAINEE"
560 C="":PRINT@979,"ANY KEY TO CONTINUE";:PRINT@0,"";:SOUND11,11
:C=INKEY$:IFC=""THEN560ELSECLS
```

Choose the number of guesses that the player will have, and determine where the bomb will be placed.

```
570 RANDOM:N=RND(4)+2:F=RND(7):R=RND(9):G=0:CLS:Q=0
```

Build the hotel on the screen.

```
580 PRINT@27,"BUILDING";:SOUND90,100
590 FORX=1TO7:PRINT@960-128*X,A(X);:PRINT@1020-128*X,A(X);:PRINT
@0,"";:SOUND45,200:NEXTX
600 Q=1019:FORX=1TO7:Q=Q-128:FORY=1TO9:PRINT@Q-6*Y,A(10);:PRINT@
0,"";:SOUND10,100:NEXTY:SOUND255,15:NEXTX
610 PRINT@27,"            ";:FORX=1TO9:PRINT@6*X,X;:PRINT@0,"";:SOUN
D10,50:NEXTX:Q=1
```

Display the number of attempts left and get the player's input.

```
620 PRINT@991,"PIAZZA HOTEL  ATTEMPTS LEFT =";N;:PRINT@0,"";:SOU
ND200,20:SOUND100,50:IFN=0GOTO770
630 C="":PRINT@960,"FLOOR (1-7)?";:C=INKEY$:IFVAL(C)<1ORVAL(C)>7
THEN630ELSEG=VAL(C):PRINTG;:PRINT@0,"";:SOUND100,60
640 C="":PRINT@976,"ROOM (1-9)?";:C=INKEY$:IFVAL(C)<1ORVAL(C)>9T
HEN640ELSEH=VAL(C):PRINTH;:PRINT@0,"";:SOUND150,45
 650 PRINT@837-128*(G-1)+(H-1)*6,A(9);:PRINT@0,"";:FORX=1TO10:SOU
ND255,2:SOUND150,4:SOUND200,3:SOUND100,5:NEXTX:IFG=FANDH=RTHEN10
 40
```

Branch to the proper "hint" subroutine.

```
660 IFG<>FTHEN690
670 IFH<R,X=11:GOSUB1030
680 IFH>R,X=12:GOSUB1030
690 IFH<>RTHEN720
700 IFG<F,X=14:GOSUB1030
710 IFG>F,X=13:GOSUB1030
720 IFH<RANDG<F,X=15:GOSUB1030
730 IFH<RANDG>F,X=17:GOSUB1030
740 IFH>RANDG<F,X=16:GOSUB1030
750 IFH>RANDG>F,X=18:GOSUB1030
760 GOTO620
```

Explode the bomb.

```
770 Z=837+6*(R-1)-128*(F-1):FORX=1TO10:PRINT@Z,A(8);:PRINT@0,"";
:SOUND255,10:PRINT@Z,A(10);:PRINT@0,"";:SOUND10,100:NEXTX
```

```
780 L=47:X=2:GOSUB1000
790 L=81:X=5:GOSUB1000
800 L=203:X=7:GOSUB1000
810 L=325:X=9:GOSUB1000
820 L=453:X=9:GOSUB1000
830 L=581:X=9:GOSUB1000
840 L=715:X=7:GOSUB1000
850 L=849:X=5:GOSUB1000
860 SOUND255,100
870 GOSUB1010
880 PRINT@0,"";
890 FORE=50TO75STEP.1:SOUNDE,1:NEXTE
900 PRINT@465,A(19);:PRINT@471,A(20);:PRINT@477,A(20);:PRINT@483
,A(21);:PRINT@489,A(22);:PRINT@0,"";:SOUND255,250
910 GOSUB1010
920 PRINT@340,"TOO BAD! ---- TOO LATE!";:GOSUB1020
930 PRINT@404,STRING$(23,"=");:GOSUB1020
940 PRINT@467,"YOU ARE NOW A DETECTIVE IN";:GOSUB1020
```

```
950 PRINT@529,"THAT BIG HOTEL IN THE SKY, TO";:GOSUB1020
960 PRINT@595,"RETURN TO EARTH FOR ANOTHER";:GOSUB1020
970 PRINT@661,"GAME PRESS ANY KEY...";:GOSUB1020
980 C="":C=INKEY$:SOUND11,11:IFC=""THEN980
990 GOTO570
```

Subroutine to draw bomb display.

```
1000 L=L-6:FORW=1TOX:L=L+6:PRINT@L,A(8);:PRINT@0,"";:SOUND100,10
:NEXTW:SOUND255,10:RETURN
```

Subroutine to clear center of display.

```
1010 L=273:FORX=1TO6:L=L+64:PRINT@L,STRING$(30," ");:NEXTX:RETUR
N
```

Subroutine to click the audio amplifier.

```
1020 PRINT@0,"";:SOUND100,10:RETURN
```

Subroutine to print a hint at the location of the player's guess.

```
1030 PRINT@837-128*(G-1)+(H-1)*6,A(X);:PRINT@0,"";:FORY=1TO10:SO
UND255,2:SOUND(X*18),4:SOUND200,3:SOUND(X*7),5:NEXTY:PRINT@960,S
TRING$(31," ");:PRINT@0,"";:N=N-1:Q=Q+1:RETURN
```

Print the results at the end of a successful game. Determine the rating based on the number of guesses it took to find the bomb, and compute the number of guests saved.

```
1040 PRINT@837+6*(R-1)-128*(F-1),A(8);:PRINT@0,"";
1050 FORX=50TO10STEP-1:SOUNDX,11:SOUNDX+45,10:SOUNDRND(55),9:NEX
TX:L=-64:FORX=1TO7:L=L+128:FORY=1TO9:PRINT@L+6*Y-1,A(9);:PRINT@0
,"";:SOUND30,50:SOUND20+RND(20)-10,80:SOUND10,100:NEXTY:NEXTX
1060 GOSUB 1010
1070 PRINT@343,"CONGRATULATIONS!";:GOSUB1020
1080 PRINT@407,STRING$(16,"=");:GOSUB1020
1090 L=RND(100)+(600-Q*100):PRINT@466,"YOU HAVE SAVED";L;"GUESTS
.";:GOSUB1020
1100 PRINT@528,"YOU ARE NOW A ";B(Q);
1110 PRINT@657,"PRESS ANY KEY FOR ANOTHER GAME.";:GOSUB1020
1120 GOTO980
```

Print the instructions.

```
1130 PRINT:PRINT
1140 PRINT"S-80 TRANSLATION BY MICHAEL R. FREIFELD   AUTHOR:  GA
RY DOMINICK"
1150 PRINT"     AS THE HOUSE DETECTIVE OF THE PIAZZA HOTEL, YOU
HAVE JUST"
1160 PRINT"BEEN INFORMED THAT THERE IS A BOMB IN ONE OF THE ROOM
S.  YOUR"
1170 PRINT"JOB IS TO FIND IT - F A S T."
1180 PRINT
1190 PRINT"     YOU BEGIN YOUR SEARCH BY CHOOSING A FLOOR AND A
ROOM"
1200 PRINT"NUMBER. IF YOU ARE NOT CORRECT, YOU WILL BE GIVEN CLU
ES TO HELP"
1210 PRINT"YOU NARROW YOUR SEARCH.  YOU NEVER KNOW WHAT YOUR NEX
T GUESS"
1220 PRINT"WILL LEAD TO.  SOMETIMES IT'S THE CORRECT ROOM.  SOME
TIMES"
1230 PRINT"IT'S. . . B O O M !"
1240 RETURN
```

TRS-80® SWAT TABLE FOR:
PIAZZA HOTEL

LINES	SWAT CODE	LENGTH
10 - 80	TJ	528
90 - 160	PN	530
170 - 260	YJ	526
270 - 350	DA	529
360 - 450	OF	511
460 - 540	GT	558
550 - 610	IQ	519
620 - 730	QS	524
740 - 850	KN	315
860 - 970	VM	420
980 - 1050	KU	522
1060 - 1170	RK	489
1180 - 1240	KA	309

Tasmon

A review by Tim Knight

By Bruce Hansen. From The Alternate Source, 704 North Pennsylvania Ave., Lansing, MI 48906 (517)482-8270. For TRS-80® Models I and III with disk or tape. Suggested price $29.95.

In my experience with Machine Language and Machine Language programming utilities, I've seen quite a few "*T-BUG*" type monitors. Some are barely adequate, some are good, but I've just bought a monitor that I wouldn't hesitate to call incredible! This monitor, "*TASMON*" (for "The Alternate Source MONitor"), would certainly suit the needs of almost any Machine Language programmer. *TASMON* will work on the TRS-80® Model I or Model III with either tape or disk.

After I ordered *TASMON*, it arrived promptly. Much to my surprise, it contained a 45 page manual (with a complete table of contents), a handy reference card and a registration card which I turned in so that I could be notified of any updates in *TASMON*.

I was disappointed to find that I simply couldn't load the program. The Alternate Source is a fine company, but I'm afraid that their tape duplication procedures aren't the most reliable. However, I flipped the tape from the high baud (Model III) side to the low baud (Model I) side, and the program loaded.

TASMON has many nice features. I'm afraid I won't be able to list all of them, but I'd like to point out some of the important "pluses" in the program. To begin with, the documentation is very complete, and a short machine language program (TEST/CMD) is included to get the user started.

The main purposes of any monitor are: (1) to modify and/or replace the registers of the Z-80 (or other memory areas), and (2) to manipulate a Machine Language program in order to make it work more effectively; or just to make it work! *TASMON* performs these functions quite effectively.

Since *TASMON* can directly modify any part of accessible memory, you can move "up" and "down" in memory (byte-by-byte) examining the contents of any one byte. If you want to see more than one byte, you simply initiate a "memory" dump of a full 15 bytes to the screen, then you can move the memory observation area up or down. You may also "jump" to any location in memory to observe what is there.

Disassembly is also easy to implement with *TASMON*. The disassembled listing of the Machine Language program in memory may be printed out either on the screen or on the printer.

Another useful feature is a built in "hexadecimal calculator." With it hexadecimal numbers may be added or subtracted very simply. This makes programming a bit easier for those of us who can't easily do hexadecimal arithmetic.

One of *Tasmon*'s features that is especially handy and unique is the user-definable function. Pressing one key ("U") initiates a function that is defined only once.

Input and output are supported with *TASMON*. Output of disassembled listings, loading and saving of programs, etc. are all simple to do. The entire screen may be saved in memory, or it may be saved by printing it out on the printer.

The ability to "view" a program is also available. It allows you to examine the starting and ending addresses of a particular program, in addition to the filename and entry point.

Breakpoints are important to a monitor program, since the user might want to stop a Machine Language program in one particular place to examine what has occured in the registers up to that point. *TASMON* gives the user the power to set from one to nine breakpoints and the ability to "single step" a Machine Language program or to trace a program at any one of eight different, user selected speeds.

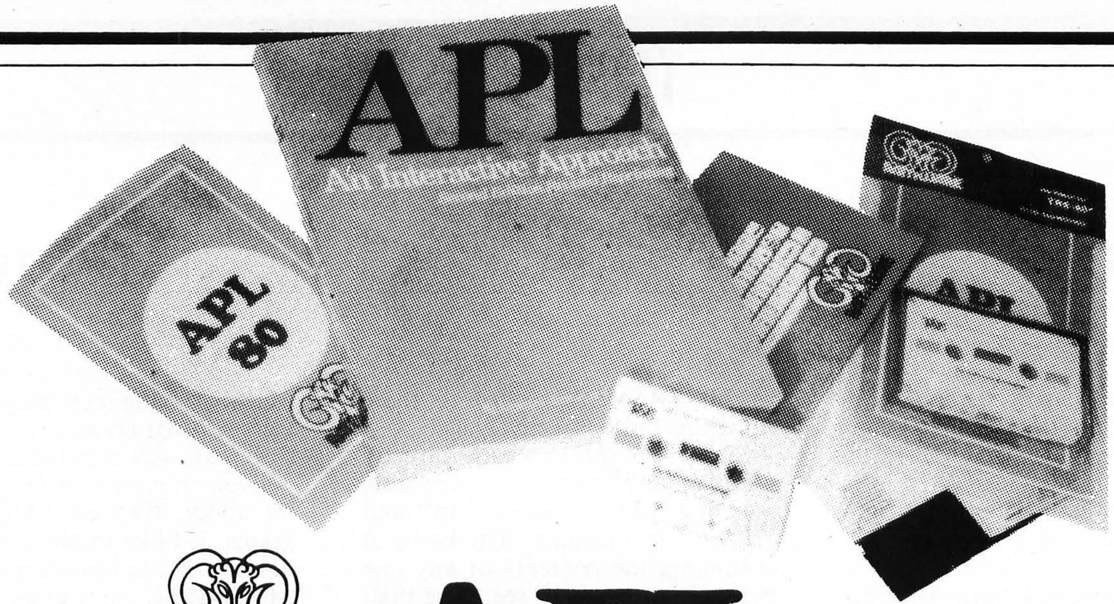
Even though I've been talking a lot about Machine Language programs, I'd like to make it clear that *TASMON* can also view the actions of BASIC programs. It is fascinating to watch the BASIC interpreter in action.

I think *TASMON*'s power speaks for itself. I tried this program because it was mentioned very briefly in *80-Microcomputing* as being an excellent monitor. I'm not sorry that I bought it, and I hope that my review will encourage others to try *TASMON*. The program is sophisticated and well thought out. The commands are very well designed and the documentation is thorough. The reference card is a nice addition. It is a handy, often used source of information. I get the feeling that the author, Bruce Hansen, wanted to create a better monitor and I believe he has succeeded; I congratulate him.

TASMON's price is a little higher than that of most other monitors, but since *TASMON* is the most powerful TRS-80® monitor I have ever seen, it is, in my opinion, well worth the money.

I can't recommend *TASMON* highly enough, but I do have one complaint — the tape. It's annoying to constantly get bad tapes from The Alternate Source. Unfortunately, their tape duplicating procedure is faulty and should be corrected. I have heard that Charles and Joni at The Alternate Source are already working on this. With the great programs, such as *Tasmon*, that they are marketing, The Alternate Source should sell their customers dependable program copies.

Every serious TRS-80® Machine Language and BASIC programmer should consider purchasing *TASMON* to become a part of their "tool box." It's usefulness should prove it to be well worth the price.



RAMWARE

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

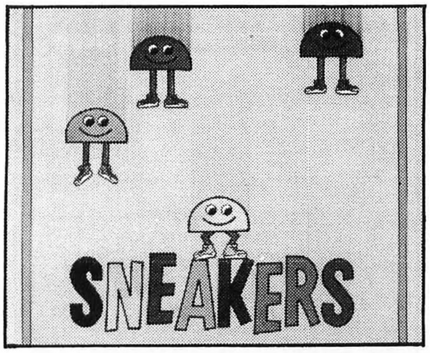
Reduced feature: 16K Level II tape version, no lessons.
Transpositions, format, and inner product not implemented. Reduced domain for some functions, 6 digit accuracy.
....$14.95 on cassette

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, domino, and matris 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 603-673-0585

NEW PRODUCTS

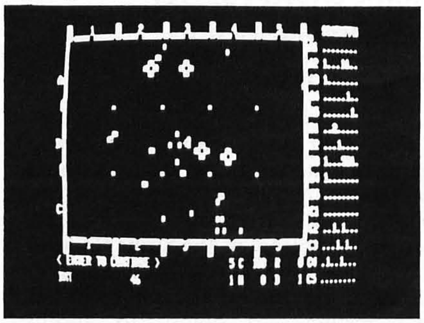


**SIRIUS SOFTWARE, INC.
10364 Rockingham Drive
Sacramento, CA 95827
(916) 366-1195**

It's now evident that *Sneakers* is one of the most popular Apple computer games ever! Players and critics alike concede that *Sneaker*'s popularity is based on it's seemingly endless variety of challenges. The game features eight separate attacks from some of the most animated and colorful characters to ever attempt to stomp you into submission. Get ready for wave after wave of Sneakers, Cyclops, Saucers, Fangs, H-Wings, Meteors, Scrambles and Scrubs. There are five levels of difficulty, making *Sneakers* forty games in one!

Sneakers requires an Apple II or II+ computer with 48K and one Apple disk drive and is playable with keyboard or Apple-compatible paddles and joysticks. *Sneakers* was written in assembly language by Mark Turmell. *Sneakers* is now available at your local Apple dealers or distributors, or you can order direct from Sirius Software. Suggested retail price is $29.95.

Hypergate Centurion



**SYNERGISTIC SOLAR, INC.
P.O. Box 560595
Miami, FL 33156**

Hypergate Centurion and *Hypergate Patrol* are two totally original games for the 16K TRS-80® Model I and Model III microcomputers.

You are the Hypergate Centurion, the guardian of the gateway to other galaxies. It is your sworn duty to protect the friendly military and cargo spaceships as they transit the hypergate. You carry out your mission nestled deep within the core of the Centurion Base Asteroid. You have at your disposal the centurion computer, a powerful plasma disruptor weapon, and a small but efficient fleet of patrol ships.

This is not a simple hand-eye coordination shoot'um down, but a highly sophisticated real time simulation requiring learned mental skills of decision and judgement.

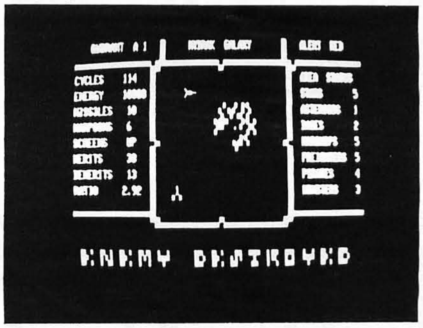
Some features of *Hypergate Centurion* are: a cast of fifty different objects displayed in detail; four skill levels, from trainee to master; five quarrelsome alliances; and a detailed, thirty-two page instruction manual.

In *Hypergate Patrol*, you command a Hypergate Patrol Ship. You must contend with most of the same situations from *Hypergate Centurion* on a grueling ship to ship basis. The Centurion on Duty assigns you various missions; from rescuing stranded spacecraft to harpooning huge space monsters. You must also master the technique of hypergate navigation so that you can patrol the galaxies of the Local Super Cluster.

This is an intense, one-player, stop-action strategy simulation. It is written in BASIC but uses Machine Language subroutines for the explosion scenes and sound effects. *Hypergate Patrol* contains excellent action sounds and action graphics, four weapon types, numerous command options, and a multitude of combat and navigational aids.

Both games are sold as a set on one cassette tape. Disk compatible versions are also available. The cassette tape retails for $39.95.

Hypergate Patrol



**C & H VIDEO
110 West Caracas Avenue
Hershey, PA 17033
(717) 533-8480**

Now you can create effective slide show presentations for a variety of uses with C & H VIDEO'S *The "Slide" Show,* utilizing High-Res graphics and 20 different, specially created transitions between slides.

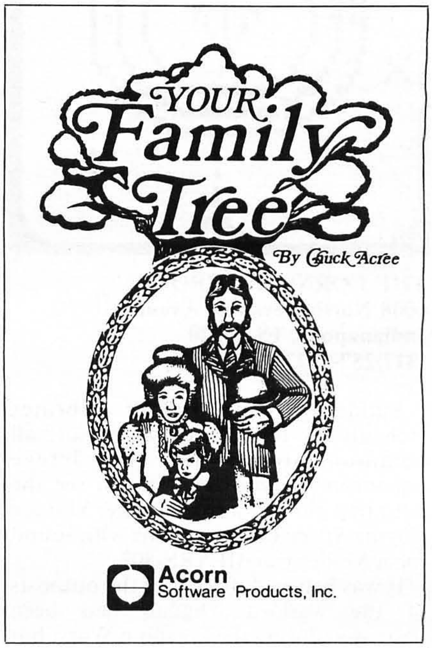
Great for business presentations, educators, exhibits, lectures, etc.. The program has many features including automatic advance, time settings, slide advance mix option, and the 20 transitions. It can contain up to 75 slides which can be repeated.

The "Slide" Show works with a 48K Apple in DOS 3.3 with one or two disk drives and is not copy protected. It is compatible with NTSC TV signal processing hardware. It can be ordered directly from C & H VIDEO and the retail price is $49.95.

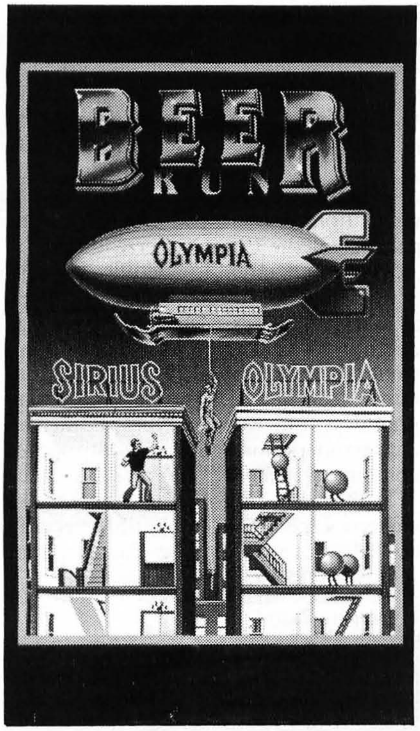
**ACORN SOFTWARE PRODUCTS,INC.
634 North Carolina Ave., S.E.
Washington, D.C. 20003
(202) 544-4259**

Your Family Tree is a comprehensive, genealogical program that lets you avoid the rigidities of a paper-based family tree. It quickly and easily sets up a database to hold pertinent information about each ancestor including: name, date and place of birth, marriage and death information, and a comment line. Access to information in *Your Family Tree* is virtually unlimited. You have full search capabilities on any key field using full or partial information.

Your Family Tree is a genealogy program for the Radio Shack TRS-80® Model I and Model III microcomputers. It is available on Model I/III compatible cassette (16K) or diskette (32K). Model III owners with two disk drives use the CONVERT utility. This program is priced at $29.95.



NEW PRODUCTS

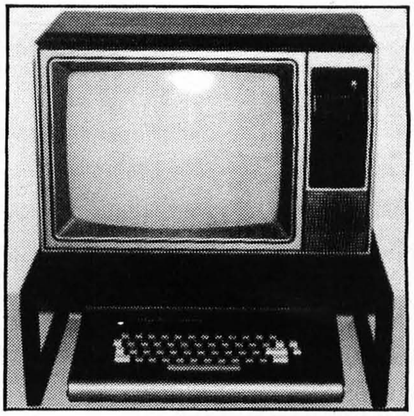


SIRIUS SOFTWARE, INC.
10364 Rockingham Drive
Sacramento, CA 95827
(916) 366-1195

Beer Run is a delightful, light-headed game designed to bring you many hours of refreshing fun. The object of *Beer Run* is to catch Artesians. You are a person at the bottom of the Sirius Building. Your task is to look for Artesians by climbing up the Sirius Building, then transporting, via blimp, to another building and climbing back down to the ground. Ladders and elevators are used to climb. Guzzlers and Bouncers chase you in both buildings.

Written in assembly language by Mark Turmell, *Beer Run* features easy control through keyboard, Apple-compatible paddles and joysticks, or Sirius Software's Joyport^{TM} with ATARI®-type joysticks, as well as escape feature for temporary halt of game, restart, sound toggle on/off, and several different levels of play. *Beer Run* utilizes Apple's color, and high resolution graphics at their best.

Beer Run operates on an Apple II or II+ computer with 48K of memory and one Apple disk drive. *Beer Run* is now avaliable and can be ordered directly from Sirius Software or through your local dealer or distributor. Suggested retail price is $29.95.

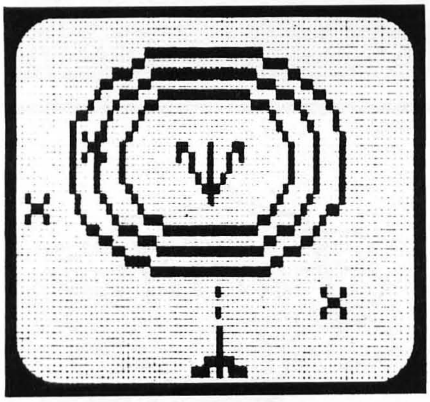


ADVANCED EFFORT-SAVER PRODUCTS, INC.
P. O. Box 5001
Hialeah, FL 33014
(305) 821-9961

The *Desk Top Video-Printer Stand* was designed to "save space" and "save money" by converting the microcomputer user's present desk or table into a specialized, computer work-station. Used as a video stand the user's TV sits on the *Desk Top Video-Printer Stand* and the microcomputer conveniently slides underneath. Work space is considerably increased and the video screen is raised to eye level.

If used as a printer stand, the user's disk drives, tape recorder, other devices or fanfold paper slide under the printer which rests on the *Desk Top Video-Printer Stand.* This further increases usable desk top space while placing all components within easy reach.

The *Desk Top Video-Printer Stand* is compatible with the TRS-80® Color Computer or VIDEOTEX, ATARI® 400 or 800, Apple II or III, TI 99/4 or any other Micros/Terminals with detachable keyboards. Most small to medium sized printers such as the MX-80, MX-100 or line Printer VIII work well. The *Desktop Video-Printer Stand* features sturdy metal construction; colors available are black or beige and dimensions are 19.5"W-12" D=6.75"H. It sells for $39.95.



THE CORNSOFT GROUP
6008 North Keystone Avenue
Indianapolis, IN 46220
(317)257-3227

Suddenly your fighter vibrates violently. A force field has cut off all communication with the Jeteye Squadron. In your viewer you see the pulsating glow of a space castle. You are playing *Space Castle*, a game with sound for a Model I or III TRS-80®.

It was believed that all of the outposts of the warlord Yugdab had been destroyed during the Freedom Wars, but the space castle before you is no illusion. As you focus your weapons on the orbiting armor shields, you realize that more is at stake than your puny existence. You do, however, place confidence in the ability of The Source to restore your life based on your performance as a Freedom Fighter. If you should eventually fail against Yugdab your point score is your only satisfaction.

Be aware of Yugdab's Intelligent Mines that patrol the orbiting shields. Once they leave the safety of the space castle, they will hunt you down like a Namelran Fox Droid.

Due to the intense radiation emanating from the space castle, your short-range viewer is inoperative. Consequently, you will have to coordinate the battle by watching the transmission from your scout satellite, Kemosabe.

Yugdab has been known to attack only when his inner armor shield has been penetrated and he is never destroyed. He can immediately resurrect himself to continue the battle. However, the reward for making Yugdab enter another life system temporarily is another ship with which to do battle and 500 points added to your score.

DP DIRECTORY
P. O. Box 562
Bloomfield, CT 06002

DP Directory, a new data processing reference magazine, publishes the Tables of Contents of over 100 data processing periodicals each month. In addition to *SoftSide, DP Directory* covers dozens of data processing magazines dealing with hardware, software, systems development, telecommunications, graphics, word processing and personal computing. 12 monthly issues are available for $48.00 from *DP Directory.*

NEW PRODUCTS

COMPU-MATE CORPORATION
6305 Arizona Avenue
Los Angeles, CA 90045
(213) 991-7098

Intelligent interface modules from Compu-Mate will enable the ATARI® 800 and 400 personal computers to accomplish many additional personal and business applications.

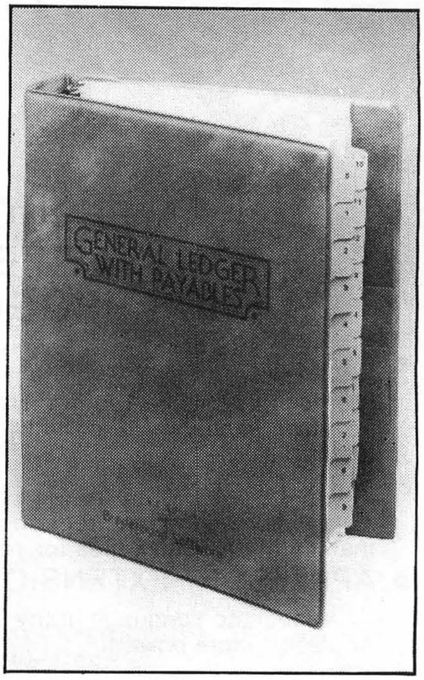
The *Model CM-1000 Printer Interface* includes one EIA standard serial port (standard synchronous protocol) and one 8 bit parallel port (Centronics compatible). Both of these ports are for use with standard serial or parallel printers. The unit comes with a simple customization program that allows the user to tailor special control codes for each port. This allows the unit to be used with many popular printers. Additional features include: Standard baud rates from 300 to 38,400, bi-directional communications for use with modems etc., software selectible port addressing, automatic powerup compatiblility with the ATARI® 825 printer on the parellel port and Diablo compatible printer on the serial port.

The *CM-1000/V Printer Interface* includes the CM-1000 unit described above plus an 80 column video display generator. This generator enables the 800 and 400 computers to perform many functions such as full page width word processing and other tasks requiring an 80 column instead of a 40 column display. Additional features include: full 96 ASCII character set, upper and lower case characters with below the line descenders, direct connection to the 800 and 400 video monitor ports, reverse video, software downloadable character set, and compatibility with any video display capable of 80 columns.

Features common to both interfaces include: no required physical changes to computers, both interfaces connect directly to the computers' standard serial I/O port, no cables to become disconnected from the computer, both operate under DOS 2.0S and future DOS versions, and both have a one year warranty. The *CM-1000 Printer Interface* is priced at $289.00. The *CM1000/V Printer Interface* is priced at $489.00. A kit (Model CM-10/V) to upgrade the CM-1000 to the CM-1000/V (adds video display generator) is available for $225.00.

In an effort to inform our readers of new products, ***SoftSide*** welcomes your company's input to this section. Send all information to:

***SoftSide* Publications**
6 South Street
Milford, NH 03055



BRODERBUND SOFTWARE, INC.
Business Software Division
1938 Fourth Street
San Rafael, CA 94901
(415) 456-6424

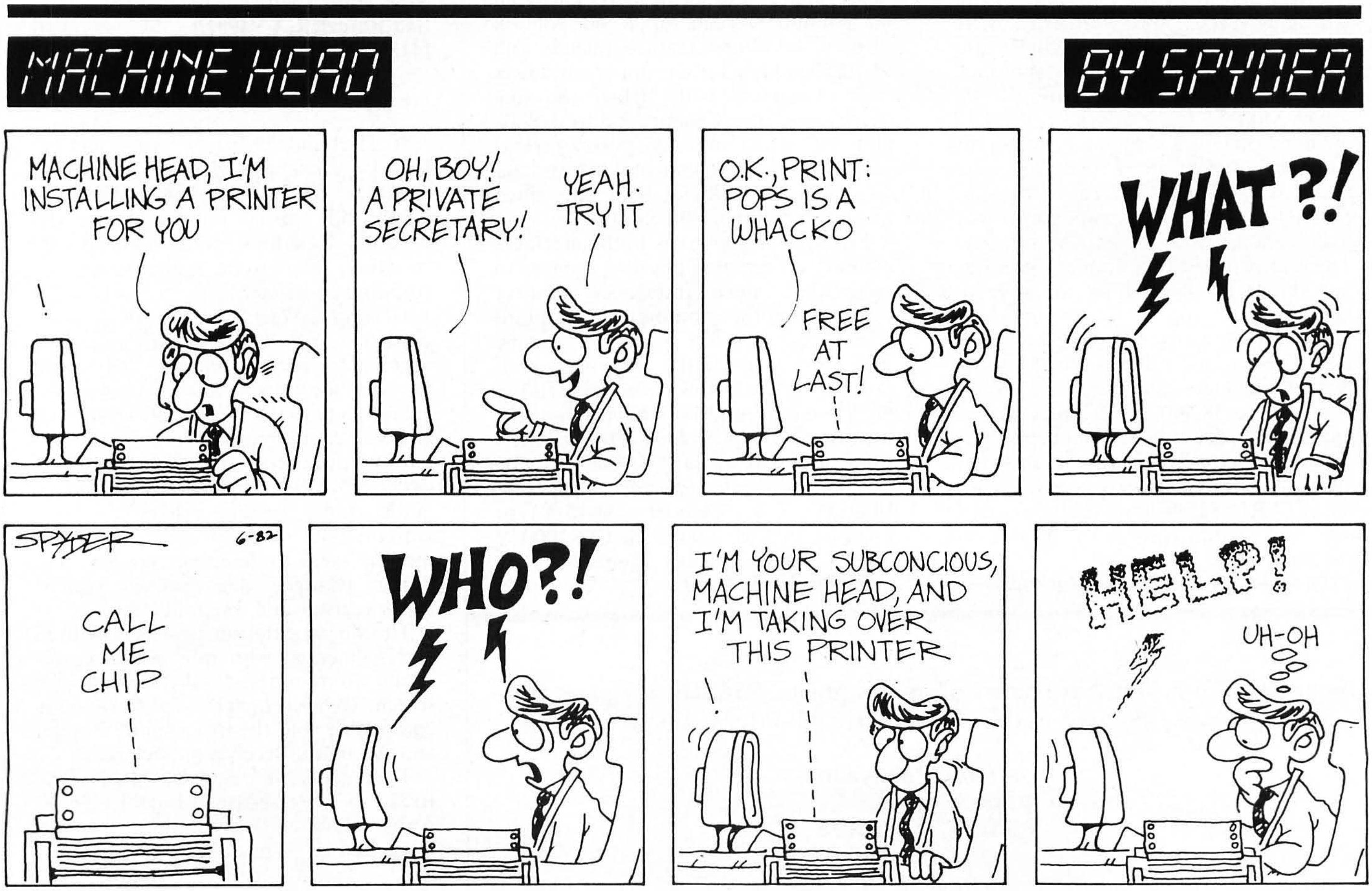
Broderbund Software, Inc. has released a *General Ledger with Payables* program, by Hal Faulkner, which will handle all normal journal ledger and payable functions for financial accounting, plus some management accounting functions.

General Ledger with Payables has a capacity of 2000 ledger accounts, unlimited journal entries, 200 open payable accounts, 1900 open invoices, values up to $9,999,999,999.99, 1000 checks per disk and user definable account numbers up to 10 digits (9999.255.255). It prints checks, journal audit trails, balance sheets, income statements with budgets, history, month, year-to-date percentage and dollar change, department reports, check register and much more.

The documentation is clearly written and segmented with tabs which correspond to numbered functions on the screen. *General Ledger with Payables* is compatible with the Broderbund Payroll and Accounts Receivable packages.

This program requires 64K Apple II/II+ with 16 sector disk controller or Apple III, two disk drives, and a printer. Suggested retail price is $495.00.

Advertiser's Index


MACHINE HEAD
BY SPYDER
MACHINE HEAD, I'M INSTALLING A PRINTER FOR YOU
!
OH, BOY! A PRIVATE SECRETARY!
YEAH. TRY IT
O.K., PRINT: POPS IS A WHACKO
FREE AT LAST!
WHAT?!
SPYDER 6-82
CALL ME CHIP
WHO?!
I'M YOUR SUBCONCIOUS, MACHINE HEAD, AND I'M TAKING OVER THIS PRINTER
HELP!
UH-OH

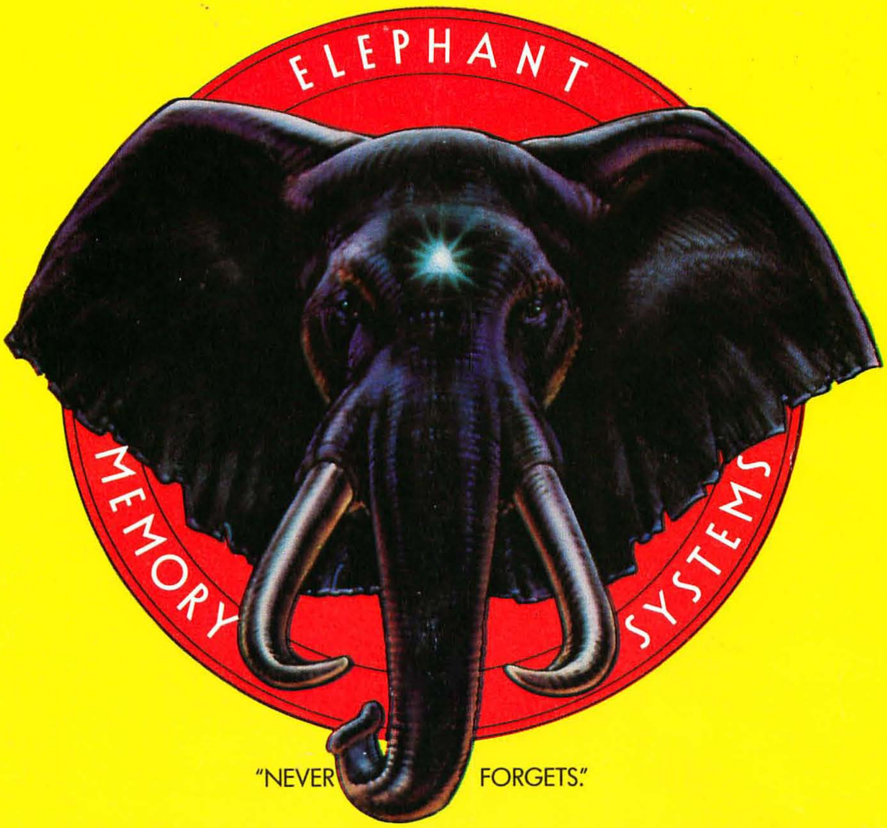
BOUND TO PLEASE: SoftSide's Deluxe Edition

Twelve issues of SoftSide — October 1980 to September 1981

SoftSide is pleased to announce this limited edition: twelve issues sewn into blue buckram. This sturdy library bound collection will be available in limited quantities, so order yours now.

The SoftSide Deluxe Edition — $75.00
Send a check or use your MasterCharge/VISA credit card.
To order write: **SoftSide Publications**
6 South Street
Milford, NH 03055

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

Distributed Exclusively by **Leading Edge Products, Inc.**, 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.