

SoftSide™

February 1982 Vol. 5, No. 5 THREE DOLLARS



HELP WANTED

Man for man, the Israeli army and Napoleon's Grande Armée are two of history's finest fighting forces. SSI's latest games recreate both in meticulous detail — complete in every facet except one: We've left them leaderless. Now we need someone to take the helm of command — someone like you.

SOUTHERN COMMAND™ Job Description

Superb color Hi-Res graphics reproduces the setting for this battalion-level game which recounts the Israeli counterattack to cross the Suez Canal during the October War of 1973 against Egypt.

The 28-by-39 hex-grid map of the Sinai can be viewed as one strategic screen or in twelve



\$39.95

separate screens using scrolling.

As the Israeli commander, you have to smash past enemy strongholds, cross the Suez, and establish a bridgehead. In order to accomplish this, your armor, artillery and infantry units — along with your airstrikes — must successfully protect the slow-moving bridging units as they push towards the Canal.

The Egyptian commander's goal is to stop your advance using the

forces at his disposal, which include the potent SAM missiles. His air force can be called upon to negate your aerial threat.

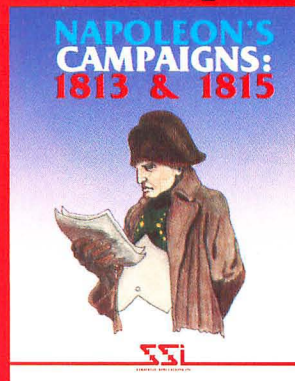
To reflect history accurately, Egyptian and Israeli forces differ in efficiency level and strength points. A unique "delayed move" feature allows for ambushes with infantry and artillery.

Thanks to machine-language programming, the computer can rapidly and efficiently calculate, display, and implement combat results to give you a fun and fast-moving game.

You never need to worry about being out of action for lack of a playing partner. The computer can direct the Egyptians and play you at any one of four levels of difficulty. So whether you're a novice or veteran gamer, you're guaranteed a challenging match.

If you feel you have what it takes (and don't wish to relocate in the Middle East), get SOUTHERN COMMAND today and the job is yours!

NAPOLEON'S CAMPAIGNS: 1813 & 1815™ Job Description



\$59.95

This advanced-level, board-assisted computer simulation presents a leadership opportunity where experience in strategy gaming really helps. Aside from fully appreciating the painstaking detail and design efforts we've put into the game, a hardcore strategist can more effectively deal with the complexity, frustrations, and uncer-

ainties inherent in Napoleonic warfare.

Leipzig and Waterloo are the two campaigns featured. Like Napoleon, Schwarzenberg or Wellington, all your com-

mands are sent to your troops and information about them received via dispatch. Knowledge of troop positions and estimates of enemy strength are only as good as your reconnaissance patrols.

The computer plays the individual corps commanders, whose leadership ratings have been preprogrammed based on historical data. How your orders are carried out depends entirely on the corps commanders, who may follow them to the letter or do so with hesitation. Misinterpretation and even outright disobedience of your directives are also possible.

Night, rain, and terrain all affect troop movement. Intricate rules that deal with the effects of fatigue, corps morale, and leadership on battle outcome serve to mirror history faithfully. They also complicate your decision making and strategy planning.

We know we've painted a pretty tough picture of this job, and we don't expect everyone to apply for it. We're looking for those who can meet the challenge and overcome the obstacles. For these people, we guarantee the same feeling of gratification the Emperor himself often felt when he added up his victory points.

Two-player and solitaire scenarios are provided for both campaigns.

Our Want Ad has all the information you need to land this great job.

Dept 105, Ste 610, SF 94111. EOE

BATTLE COMMANDER WANTED

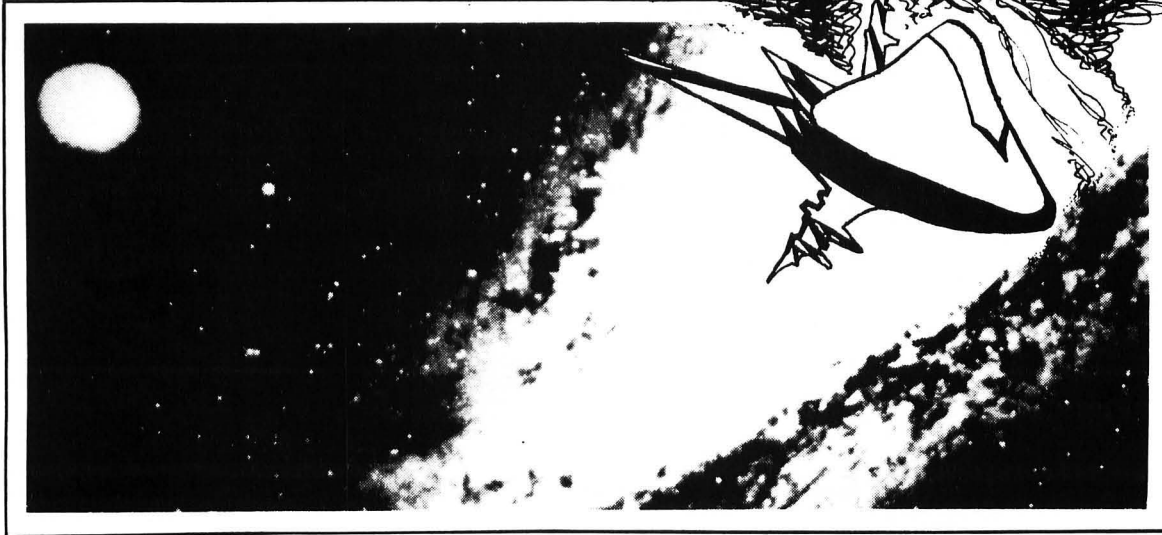
Two of history's finest armies need resourceful, daring, decisive leader to defeat enemies of superior size and strength. Experienced strategist desired but not necessary. Must have 48K Apple II computer with ROM card or Apple II Plus and 1 mini floppy disc drive. Great benefit package of fun and excitement. Flexible hours. Secure job with no-cut contract. Immediate opening. Applications accepted today at your local game/computer store. VISA and M/C card holders, apply directly by calling toll free 800-227-1617, ext. 335. In California, call 800-772-3545, ext. 335. Or write to: Strategic Simulations Inc, 465 Fairchild Dr., Suite 108, Mountain View, CA 94043.

STRATEGIC SIMULATIONS INC

PROGRAMMER
3 to 5 years programming exp. Working

PROGRAMMER/ANALYST

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



RAINWARE PRESENTS...

Time Lord

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

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

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

And once you find The Master can you destroy him?

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

**SoftSide
Selections**

6 South Street Milford NH 03055

PUBLISHER
Roger W. Robitaille, Sr.

ASSOCIATE PUBLISHER
EDITOR
Randal L. Kottwitz

PROGRAMMING EDITOR
Jon Voskuil

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

ART DIRECTOR
PROMOTION MANAGER
Nancy Lapointe

ADVERTISING MANAGER
Clif Campbell

PRODUCTION MANAGER
Lynn Wood

PRODUCTION DEPARTMENT
Lynda Fedas
Karen Lawrence
Tom Stanton

ADMINISTRATIVE ASSISTANT
Nancy Macy

CUSTOMER SERVICE
Jean Matthews

DEALER SALES
Kathie Maloof

STAFF
Jerry Butler
Donna Jean
Doris Miller
Cindy Schalk

Photographs by Dean F. H. Macy

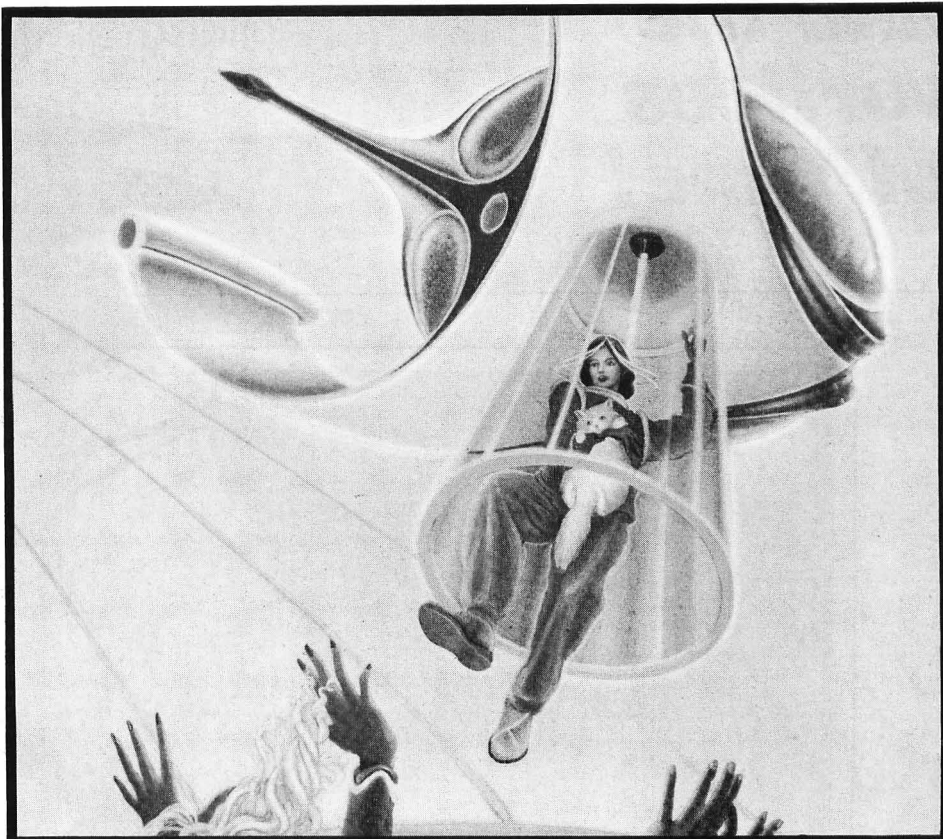
SoftSide is published each month by *SoftSide* Publications, 6 South Street, Milford, New Hampshire 03055. Telephone 603-673-0585. Second class postage paid Milford, New Hampshire and pending at additional mailing offices. ISSN: 0274-8630. Subscription rates: USA and Canada, \$30.00/12 issues. USA and Canada First Class APO, FPO, Mexico, \$40.00/12 issues. Overseas air mail: \$60.00/12 issues. Media subscription rates: Magazine and cassette, \$75.00/12 months. Magazine and disk, \$125.00/12 months. APO, FPO, Canada, Mexico, (add), \$50.00/12 months. All remittances must be in U.S. funds. Mail subscription inquiries to *SoftSide* Publications, 515 Abbot Dr. Broomall PA 19008. Entire contents © copyright January, 1982, *SoftSide* Publications. All rights reserved.

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

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

TRS-80®, Apple, and ATARI® are registered trademarks of The Tandy Corporation, The Apple Computer Company, and Warner Communications, respectively. Envyrn, Envyrnment, Envyrnese, and diversions thru Envyrn are registered trademarks of Roger W. Robitaille, Sr.



Cover illustration by Bill Giese

Front Runner

25

Space Rescue

by Matt Rutter
You have been selected as a member of the exploration party searching for life on the planet Arcturus III. Radar indicates a meteor storm headed straight for that solar system. Can you, in your two-person rocket, rescue the people stranded there without crashing into a meteor? You are their only hope.

Features

10

Entertainment Tomorrow

by Allen L. Wold and Fred D'Ignazio
In another peek into the future, the authors tell how computers could help enhance the tension and excitement of fantasy role-playing games. With the addition of low-level lasers, computerized fencing would take on an unequalled element of realism.

15

Sensuous Programmer

In this tenth installment, "J" discusses the use of logical operators and relational operators in BASIC programming.

18

My Side of the Page

by Lance Micklus
In Getting a Bit Serious — Part Seven, Lance discusses marketing projections and customer support problems. See how you can profit through his experience.

23

Hardware Corner

by Edward E. Umlor
The series on disk drives continues. Read about floppy drives, floppy drives and aspects of a variety of DOSs.

80

Machine Head

by Spyder Webb

Special Features

5

DOSPLUS — How It's Different

Since the operating system of the TRS-80® DV has changed from TRSDOS to DOSPLUS, those subscribers should find this summary of the major differences helpful.

8

Comments from Tandy

by Ed Juge

In a letter to the publisher from Ed Juge, director of computer merchandising for Tandy Corporation, some of Tandy's policies about why only certain software and hardware products are supported by Radio Shack are explained.

38

Let's Be Civilized

by Leonard Buchanan

Have you learned how to react in a civilized manner to that OTHER computer your acquaintances own? Have you been able to explain your feelings about YOUR computer to others? If not, try getting civilized.

39

Reader Survey

Each month we spend a great deal of time selecting, translating, and illustrating the programs and articles in *SoftSide*. This month we are asking you to complete and return the survey found in the center of this issue. Your responses will help determine future selections of programs, articles and other features. Please take the time to help us provide you with what interests you the most.

Departments

4 Editorial

4 Calendar

6 Input

7 Hints and Enhancements

9 Outgoing Mail

APPLE™/SIDE

44 Enhanced Disk Version

Andorra by Brent Packer
Lead the country of Andorra in this simulation.

46 K-Byter

Pick-Up by William Pu

47 Program

Rubicube by F.J. Condo
This color-graphics simulation will help you solve that famous 3-D puzzle.

56 Review

Hi-Res Secrets by Cary W. Bradley

ATARI®/SIDE

59 K-Byter

System Configuration Test
by Alan J. Zett

60 Enhanced Disk Version

Kismet II by Peter Kirsch
This dice game combines luck and skill as you arrange your dice scores to get the highest total.

62 Program

Defense by Greg Schroeder

67 What's New

Ramdisk by Dean F. H. Macy

69 Review

Protector by Alan J. Zett

TRS-80®/SIDE

71 K-Byter

Lites Out by Ronald and Jordan Corn

73 Enhanced Disk Version

Help Package by Rich Bouchard
A utility for getting instructions on how to use your DOS, computer, or anything else.

74 Program

Maze Sweep
by James Garon and David Bohlke
A translation of *Maze Search*, this is an arcade-style game with optional sound.

77 Article

Modify EDTASM for the Model III — Part II
by Randy Hawkins

79 Review

Parsector V by Marvin Lewis



February 2-3

Seminar on Packet Networks Key Bridges Marriott, Arlington, VA

The "X-25 Packet Network Protocol" is a seminar sponsored by *Data Communications* magazine covering the International Standard Network Protocols. Topics will include concepts and terminology, physical level, link level, packet assembly/disassembly for non-packet mode terminals, and upper layer protocols. Seminar fee is \$550. Contact: McGraw-Hill Conference Center, 1221 Avenue of the Americas, Suite 3677, New York, NY 10020

February 3-5

Home Terminal Monitoring Service Subscriber Conference Sheraton Sand Key Resort, Clearwater, FL

The service provides a variety of research products to subscribers including a newsletter, library, and access to the laboratory in Clearwater which is developing databases applicable to the home environment. The conference will deal with implementation and marketing of this system.

February 10

Invitational Computer Conference Pier 66 Hotel, Ft. Lauderdale, FL

The seminar/display will be directed exclusively to the needs of the quantity buyer of computer and peripheral equipment.

February 22-24

Federal DP Expo Sheraton Washington Hotel, Washington, DC

This is the largest show for end users in the multi-billion dollar federal government marketplace.

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

SoftSide Publications
Calendar Editor
6 South Street
Milford, NH 03055

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

by Jon Voskuil

What makes one person fascinated or even obsessed by computers, and another totally apathetic or even hostile toward them? Which chromosome determines such inclinations? What traumatic experience during formative childhood years directs a mind toward (or away from) a sympathetic relationship with a silicon chip?

Elsewhere in this issue "J" discusses the topic of logic in BASIC programming. Does this suggest a possible line of demarcation between computerist and non-computerist? Is the one a "logical" thinker, and the other an "intuitive" thinker? While it would be futile and depersonalizing to try to classify everybody into one of those two categories, it does seem that most people tend to lean toward one or the other pattern.

The "logical thinker" tends to see life in terms of rational precepts and cause-effect relationships. He tends to enjoy scientific pursuits, discovering pattern and order and logic in the structure of the universe. He also tends to see other people as simply another part of the universe, to be related to logically and understood rationally. And perhaps because he finds that such a frustrating task, he may tend to prefer the company of things to the company of people.

The "intuitive thinker," on the other hand, has tendencies in the reverse direction. He views life more in terms of aesthetics and human values than in terms of mere, cold logic. He tends to have little patience with inanimate objects which demand a great deal of time, money, maintenance, and attention to rigorous, logical procedures — and which give no human warmth in return. In his relationships with other people he tends to be less analytical and more empathetic, seeking to understand them not primarily as intellectual beings but as sentient beings.

Stereotypes? Sure. These aren't rigid personality molds that are fixed in the order of the universe. But they do describe two contrasting facets of human nature, which are present in various mixtures in each of us. Without a balance of both logical and

intuitive capacities, we would be very lopsided creatures. In fact, a person who is a 30/70 mix of intuition and logic may view a person who is a 70/30 mix as a lopsided individual. And a 5/95 or 95/5 mix might be regarded by almost everyone else as a strange bird.

Here, then, is a question of some importance for the future of computerization in human society: Is the "intuitive thinker" going to be left out in the cold by the computer revolution? Do computers require rigid, impersonal logic in order to be useful tools? Will they become more and more a divisive force, ultimately counter-productive in maintaining the humanness of human society? Will aesthetics and intuition and other vital parts of our humanity be slighted and submerged by technology?

This is a fear shared by many people. Future shock has a destructive potential. The most obvious applications of computers are indeed in the realms of logic and reason. They lend themselves supremely to analysis, calculation, and deduction — the formulation of results based upon known facts and the rigid application of the rules of logic. As such they are extensions of our intellects.

Can they also become tools to extend our intuitive and aesthetic nature? Or would this really be a contradiction, an effort to reduce aesthetics and intuition to the common denominator of logic and technology? Some would say that our humanity lies ultimately in the circuits of our brains. If this were true, it might be possible one day to create a computer sufficiently complex to begin functioning at the elementary levels of human intuition. To me this precept is nonsense, and if we want the computer to function as an extension of these parts of our humanity we need to look in different directions.

What are those directions? Are they limited to using the computer simply as a new medium for musical and artistic expression? Are there other ways in which computers will come into their own as tools to promote deep human values? Or is this where we need to draw a clear line between man and machine, and not seek to combine the two?

DOSPLUS

How It's Different

The TRS-80® Disk Version of *SoftSide* now uses the DOSPLUS operating system instead of TRSDOS. There are some important differences between the two that DV subscribers should know.

1. The TRSDOS command CMD "S" is replaced in DOSPLUS by simply CMD.

2. The TRSDOS command &O, for octal number conversion, is not available in DOSPLUS.

3. DOSPLUS BASIC must be entered in a different way than TRSDOS BASIC. Instead of first typing BASIC from DOS and then answering the questions "HOW MANY FILES?" and "MEMORY SIZE?", the user must enter all information on one line from DOS, as follows:

a. BASIC — Load BASIC with zero files available and no protected memory.

b. BASIC -F:3 — Enter BASIC with three files reserved. This is the equivalent of entering BASIC from

TRSDOS and then pressing ENTER in response to the two questions. In other words, this is the same as the TRSDOS default values.

c. BASIC -M:61000 — Enter BASIC and set memory size to 61000.

d. BASIC filespec — Enters BASIC and automatically RUNs the program "filespec".

e. BASIC * — Re-enter BASIC with your program, files, and memory size intact.

Features b, c, and d can be used in combination as well. For example, from DOSPLUS you can type:

BASIC MAILIST -F:6 -M:40000

This will have the same effect as the following series of entries from TRSDOS:

BASIC
HOW MANY FILES? 6
MEMORY SIZE? 40000
RUN "MAILIST"

ATTENTION AUTHORS

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

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

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

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

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

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

Please be sure to pack your cassettes and disk carefully and to include your return address and phone number.

Send to:

SoftSide Publications
SUBMISSIONS DEPARTMENT
6 South Street
Milford, NH 03055

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

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

TRS-80 is a registered trademark of Tandy corporation.



Become a Monday Night Quarterback

NOW YOU CAN WITH THESE 2 NEW FOOTBALL GAMES FOR THE TRS-80 MODEL I AND III... ARMCHAIR QUARTERBACK AND NFL PRO-DICTOR. BOTH SELL FOR \$19.95 EACH.

PHONE (517) 754-6320
SMALL BUSINESS CONCEPTS
DEPT. 101
4710 BAYLOR CT.
SAGINAW, MI. 48604



PRO-DICTOR PREDICTS NFL GAME SCORES AND POINT SPREADS. IT SHOWS STANDINGS AND TEAM RECORDS AND MORE. IT WAS MORE ACCURATE LAST YEAR THAN JIMMY THE GREEK. FOR PLAYOFFS AND SUPERBOWL TOO!

ARMCHAIR QUARTERBACK IS A HEAD TO HEAD FOOTBALL GAME SIMULATING THE REAL THING. YOU AND YOUR OPPONENT CONTROL THE OFFENSE AND DEFENSE AND CALL ALL THE PLAYS. SOUND EFFECTS TOO!

TO ORDER INDICATE WHICH GAME(S) AND SEND \$19.95 PER GAME. WE PAY POSTAGE. PLEASE SPECIFY MODEL I (CASSETTE ONLY) OR MODEL III DISK OR CASSETTE VERSION. CASSETTE REQUIRES 16K. DISK REQUIRES 32K. FOR FREE INFORMATION OR TO ORDER CALL OR WRITE TODAY!

JOIN THE TEAM!



TRS-80® COLOR COMPUTER

Dear *SoftSide*,

I have been buying your informative magazine at my local computer center. I have been typing in the programs for the TRS-80®. This has been quite a chore for me because I own a TRS-80® Color Computer. The results of translating have been worth the frustration.

I feel I am quite lucky that I know Level II BASIC as well as Color BASIC. (Each machine has its benefits and drawbacks.) Here is my real problem: *SoftSide* seems as though it will turn to Machine Language programs. Machine Language is much faster than plain old BASIC. Unfortunately it is virtually impossible to change Level II Machine Language to Color Machine Language. This is the reason why I haven't started a subscription. Besides, most readers don't have a very good grasp of Machine Language. The Machine Language programs will only confuse them more!

Please try to stay as "Your BASIC software magazine." To all those Model I/III owners who laugh at the color "toy," just remember our games are in Hi-Res color graphics!

Paul Kerwin
New York, NY

Editor's Reply: We announced in the January issue that we will begin to support the TRS-80® Color Computer in June of this year. You can help the quality of our support by submitting your own material and encouraging friends to submit theirs. As for Machine Language, *SoftSide* remains devoted to BASIC. However, as we discover Machine Language applications we believe to be useful to our readers, we will publish them in our system-specific sections.

COPYRIGHT

Dear *SoftSide*,

I am writing you in hopes you might answer a question for me. A friend and I belong to a microcomputer club and our club has a club library. My friend subscribes to your magazine and after he has finished reading it, he makes the magazine available to the club members. We realize that most of the programs are copyrighted however would like to share them with the other club members, but do not wish to infringe upon anyone's rights.

Our question about the programs that are listed in *SoftSide* — is it permissible for one of us to type in the programs and place them in our club library, making them available to all our members ONLY?

We would appreciate any information that you might share with us on the subject. The subject of copyrights is a controversy among our club members and would make a very interesting topic at one of our club meetings.

Wesley Lyle
Battle Creek, MI

Editor's Reply: The programs that are listed in *SoftSide* are for our readers' personal use. We cannot authorize anyone to reproduce these programs for others. It would be an infringement of

our agreements with individual authors.

Authors put many hours of work into their programs and should be compensated for it. What you suggest amounts to a form of piracy.

If your members cannot or will not take the time to input the program, why should they receive the benefit of the author's hard work? Consider the author who took months to write the program. Is it fair to just give his or her work away?

A final thought — What is a program worth which is easily acquired, requires no effort, and costs nothing? The worth to you is exactly the same as the cost — nothing.

Please respect the rights of the authors. Their time, money, and efforts went into their programs and they deserve your contribution to their royalties.

NEWDOS80 RESALE

Dear *SoftSide*,

Congratulations on a fine product of yours. I own a Model I 32K TRS-80® with one disk drive and a line printer and ever since the day I purchased the disk drive, I have been using NEWDOS80 to its fullest extent. When I read about it in *Creative Computing* a few months ago, I knew that there was no way it could compare to TRSDOS. It was well worth the investment. I have found the CMD command invaluable especially in some of my applications, and in my opinion, the screen print function doubles the value of my printer, a line printer 7.

I have only one query. I have had my TRS-80® for about four years and have been writing my own application programs in BASIC for about six years. My question is about your policies on sales. For the past two years I have been considering the sale of some of my programs. The few that I had on disk that could be transfigured onto tape are not my problem; the question I ask is about my programs that use certain features unique to NEWDOS80. For instance, the CMD"xxxx command in BASIC is an illegal function call under TRSDOS. If I sold this program, it would have to be sold on a NEWDOS80 disk. I would appreciate it if you would be able to send me some information regarding your legal terms of selling your software. I regret that I do not have much legal understanding because I am only 14 years old.

Gary Katz
East Brunswick, NJ

Editor's Reply: There are three feasible solutions to your problem — all avoiding any copyright violations. First, any program may be downloaded to tape, sold to a customer with the clear stipulation that he must have NEWDOS80, then be loaded onto a NEWDOS80 disk by the customer. Second, the program may be sold on a data disk, again clearly stating the necessity that the customer have NEWDOS80. The third and probably best solution would be for you to contact APPARAT and procure the rights to sell your product with NEWDOS80. There are usually substantial quantity discounts available for commercial resale and such a combination would add retail validity to your product.

Basically, these techniques will apply to any program requiring a specific DOS. For example,

SoftSide DV is now being distributed on DOSPLUS, however, programs may be submitted on data disks, DOSPLUS disks, or cassette tapes.

TRS-80® MODEL II

Dear *SoftSide*,

I live in a remote area of Michigan and own a TRS-80® Model II. I would like to correspond with other TRS-80® Model II owners. I would appreciate it if you ran this letter in your *Input* section.

James R. Young
P.O. Box 336
Ludington, MI 49431

ROSES AND THORNS

Dear *SoftSide*,

I am not going to correct anything about your magazine or offer some far-fetched suggestions that you cannot use. All I want to do is congratulate you on your coverage of the ATARI® computer. You are the only magazine I have found that devotes nearly one-third of its contents to ATARI® computers and I'm glad that someone finally does!

I received my first issue from you free in the mail, which was back when you were in digest form and only supported the TRS-80®. (I won't say the too-common term that many people use because I feel that the TRS-80® has a lot more good points than bad ones.) My second issue again came free when you were in the expanded TRS-80®, Apple, and ATARI® version. I was both surprised and excited at all the great changes you had made since I first saw you. Later I subscribed and now almost stand by the mailbox to wait for my next issue. Keep up the good work, guys! I'll be looking forward to your coming articles on translations.

Greg Schroeder
New Port Richey, FL

Dear *SoftSide*,

I'm finally writing to a magazine for more than seeing my words in print (hopefully). First I would like to compliment you on the perfect condition of *SoftSide* when I receive it. I regret to say it doesn't remain that way for long. After many hours of reading and rereading, it does begin to look like an old pair of jeans, which I wouldn't trade for anything except maybe a disk drive.

Prior to my involvement in computers, I was an electronics hobbyist. I often wonder why *SoftSide* doesn't provide computer-related projects or hardware kits, such as a RS-232 board, or small interface kits; the market is there!

Keep up the GREAT work. Considering the pros and cons, *SoftSide* is THE best computer magazine on the market, and from the constant changes and improvements will stay number one in my book.

Thomas R. Cichowicz
Hartford, CT



Hints & Enhancements

TRS-80® SOUND

As a charter subscriber, I have received a mixture of fun and relaxation from your magazine. I have currently been involved with using sound along with some programs from your magazine. After entering the sound to some of these programs, I feel they seem more realistic. By no means am I saying the programs are not good, but the sound seems to add a nice little touch to each program. I will be showing a list of the lines I changed and/or added. I hope the other subscribers will find these changes easy to understand.

Here are the changes for *Meteor Storm*, (November, 1980):

1. First load *Meteor Storm* into the computer.
2. Add the following lines to the programs using the line numbers given.

```
10 MM$="////////////////////////":
'27 SLASHES
15 I=VARPTR(MM$): J=PEEK(I+1)+256*PEEK(I+2)
20 FORK=JTOJ+26: READX: POKEK,X: NEXT
30 IFPEEK(16396)=201POKE16526,PEEK(I+1): POKE16527,PEEK(I+2)
40 DATA205,127,10,77,68,62,1,105,211,255,45,32,253,60
50 DATA105,211,255,45,32,253,13,16,23,8,175,211,255,201
60 STOP
```

3. After typing these lines in, type RUN. Seconds after that, the STOP in line 60 will make the program break out. After READY appears, delete line numbers 20, 40, 50, and 60. The program is now ready to add sound simply by using this command:

```
X=USR( )
A number goes in the parentheses.
```

Here is the list of lines to which I added this statement.

Change the lines as follows:

```
170 IFG=0A$=INKEY$: IF A$=" "THENX=USR(1000): X=USR(1050): G=S+65 ELSEIFA$=CHR$(13)GOTO250 ELSE190
200 PRINT@S-1,STRING$(5,191);: S1$=LEFT$(S$,1): S2$=MID$(S$,2,1): S3$=RIGHT$(S$,1): S1=S: S2=S+1: S3=S+2: FORI=1TO50: NEXT: PRINT@S-1," ";:X=USR(15001)
210 S=S-5: X=USR(15000-S): ...ETC.
230 PO=PO+100: A$=INKEY$: PRINT@G-1,STRING$(3,143);: X=USR(15800): PRINT@G-1," ";: G=0: RETURN
240 PO=PO+100: A$=INKEY$: PRINT@G-65,STRING$(3,143);: X=USR(15800): PRINT@G-65," ";: G=0: RETURN
```

Of course, in order to hear the sound you need an amplifier or speaker. The type of sound that

you hear will depend on the number in the USR statement. The numbers can be changed. If this little routine looks familiar, it should. It came from none other than this magazine.

I have put sound in many other *SoftSide* programs:

Boing, (November, 1980); *Engineer*, (September, 1979); *Collision*, (July, 1979); *Missile Invasion*, (December, 1980); *Tanks-A-Lot*, (February, 1981).

I will, when time permits, be sending these and others to *SoftSide* for their readers' benefit. I would like to say a word of praise to *SoftSide* about their good magazine. I like it very much.

Richard L. Woodard
Catlin, IL

RANDOM INTEGERS

For Apple users, here is a function which will return a random integer between 1 and A. The TRS-80® already has the equivalent of this command, RND(A).

```
DEF FN RAN(A) = INT(RND(1)*A+1)
```

PRINT FN RAN(10) will print an integer from 1 to 10.

Rich Bouchard
SoftSide

MUSIC MACHINE

In the Apple program, *Music Machine* (November, 1981), the tempo of the music during playback is much too fast. The following modification was made to correct this problem:

```
8020 POK$ = "170,160,004,...etc."
```

By changing the third element of POK\$ to 004 instead of 001, this slows the tempo down by a factor of four. I found the subroutine utilized by Jon Voskuil quite conveniently added a second duration factor at \$0325 which is stored at \$0001. This modification will play the musical notes at near perfect tempo when "TEMPO = 3" is input when requested during playback.

James P. Stevens
Champaign, IL

THE FUNCTIONAL FUNCTION

Disk TRS-80®s have a very useful feature, the user-defined function (DEFFN). They are especially useful because you can pass arguments to the functions, without defining or reserving any variables. There are dozens of uses for these functions, and below are some examples.

Function MIN returns whichever argument is smaller, while MAX returns the larger.

```
1 DEF FN MIN(A,B) = (A+B-ABS(A-B))/2
2 DEF FN MAX(A,B) = (A+B+ABS(A-B))/2
```

PRINT FNMIN(3,5) would print "3" while PRINT FNMAX(3,5) would print "5".

Function HEAD\$ will clear the screen, and print A\$ centered on the top line, with CHR\$(A)'s filling in the top line of the screen. Be sure to clear enough space for the string before defining this function.

```
3 DEF FN HEAD$(A,A$) = CHR$(28) + CHR
```

SoftSide

```
$ (31) + STRING$( (64-LEN(A$))/2,A) + A$ + STRING$( (64-LEN(A$))/2+.5,A)
```

PRINT FNHEAD\$(45,"TESTING") will clear the screen, and print the word TESTING centered with dashes to its left and right.

INSS\$ is a function that will insert the string B\$ into the string A\$, at position A.

```
4 DEF FN INS$(A$,B$,A) = LEFT$(A$,A-1) + B$ + MID$(A$,A)
```

If A\$="ABCDE", B\$="123" and A=4, then after X\$=FNINSS(A\$,B\$,A) variable X\$ would contain "ABC123DE".

This last function, DICE, will print the total of A roles of a B sided die.

```
5 DEF FN DICE(A,B) = RND(B) - RND(B)*
(A>1) - RND(B)* (A>2) - RND(B)* (A>3) -
RND(B)* (A>4) - RND(B)* (A>5)
```

PRINT FNDICE(4,6) will print an integer between 4 and 24. Variable A has a maximum of 6.

Rich Bouchard
SoftSide

NEWDOS+

Here is another way to modify the JKL function of NEWDOS+ to support graphics. It is a variation of Rich Bouchard's method (November, 1981) which POKEd the changes into memory and had to be performed every time the disk was rebooted. The following changes only have to be performed once since it is the disk that is being changed.

Using SUPERZAP change disk 00067F from 3E 2E to 00 00 for printers which use standard TRS-80® codes and C6 20 for the MX-80 in non-TRS-80® mode.

For those of you who have NEWDOS+ and only one disk drive, here is a way to change the name of system disks. Since every disk must be a system disk in these systems, they all have the same name. This modification will change the name to "DISK# 00", where 00 will be replaced by the number you write on the label of the disk. All directory listings will now show this name and this should aid in filing.

Using SUPERZAP change disk 0110D0

```
from: 4E 45 57 44 4F 53 34 30
to: 44 49 53 4B 23 20 * /
```

Replace the asterisk with the first digit of the disk # + 30.

Replace the slash with the second digit of the disk # + 30.

```
Example: 1 = 30 31
         15 = 31 35
```

Stephen Milliken
Randolph, MA

For ATARI® hint see POKE YOUR ATARI® page 68

In a recent phone conversation with SoftSide publisher Roger Robitaille, Ed Juge, director of computer merchandising for Tandy Corporation, outlined some of the reasons behind Tandy's policies concerning the development of software and hardware by other companies for the TRS-80® microcomputer series. We asked Mr. Juge include his comments in a letter to be shared with SoftSide readers. We feel his answers to our questions may help enlighten anyone concerned with Radio Shack equipment.

Radio Shack®

A Division of Tandy Corporation

COMPUTER MERCHANDISING

817-390-3011

1500 ONE TANDY CENTER, FORT WORTH, TEXAS 76102

Dear Roger:

Thank you for your recent telephone call and letter. This will confirm the information I gave you over the telephone, which you asked me to put into writing so that you could pass it along to your readers. I believe we covered several subjects, so I will try to touch on most of them here.

First, on the question of why Radio Shack does not sell "Outside Software," the statement simply isn't true. Sixty percent of the software in our stores which carries our name was purchased from outside software vendors. The reason it carries the Radio Shack label is tied very closely with one of the other major questions we get quite often: "Why won't Radio Shack carry or recommend other vendor's products?" The answer is that other microcomputer companies sell through independent dealers. It is quite easy for them to recommend a product that they feel is good, because the customer purchases it from an independent dealer. That dealer (not the manufacturer) is then responsible to the customer for the proper operation and/or interconnection of that item.

Radio Shack is its own "Dealer." When a customer purchases an item from his neighborhood Radio Shack store or Computer Center, he is purchasing from Tandy Corporation. The Corporation must then be prepared on a local or home office basis to support the product we recommended. Obviously we don't allow our people to recommend products that we as a company are not able and ready to support. The risk and the liability are simply not reasonable in our environment. The question also came up about someone else's disk drives mounted in a Model III computer. In order to make our service operations as effective and as reasonably priced as possible, our people are restricted to working on our own equipment. When "Foreign" disk drives, controller, etc. are mounted inside one of our units, it simply is no longer a Radio Shack computer. Our people are not familiar with the other components nor do we have service information on them, nor do we know how well they might be expected to operate when operating properly. The result is that we simply cannot accept a unit in one of our shops for repair which contains this type of foreign component. By the same token, equipment which has been modified in any way may not be worked on, or at best will be put back into original condition at the customer's expense prior to beginning any attempted repair.

We appreciate the people who are producing software and hardware which make our machine more attractive to prospective customers. There are many excellent non-Radio Shack hardware and software products available today. There probably are some not so great products too. And there is no question that Radio Shack is not capable of producing all of the things that all of our customers could possibly want, so we do appreciate the people who are filling the gaps for us. However, we can no more be expected to guarantee someone else's software than an independent software house can be held responsible for our computer not functioning properly, and unfortunately if we recommend or sell other people's products, we must be able to stand behind them to the customer.

I am probably overlooking some points, and I knew I should have taken some notes when we spoke. At any rate, I believe I have covered the major points. I appreciate the opportunity to provide this information, and I would like also to tell your readers that we are always anxious to hear the comments and suggestions on how we might improve our service. We can't promise that we can fulfill all of the requests, but the only way we can react is to know what our customers want. Thanks again.

Sincerely,



Ed Juge
Director, Computer Merchandising



Outgoing Mail

The season of St. Valentine's Day, Presidents' Day and Income Tax Refunds is upon us. When that refund check arrives, it's refreshing to know you can go out and spice up your life a bit. *SoftSide* has chosen this to be a month of refreshment also.

In response to requests from many of you to make our multi-system format less confusing and to help prepare for the addition of our fourth system, the TRS-80® Color Computer, we are dividing *SoftSide* into computer-specific sections. Quite frankly, the task has been more difficult than we originally imagined. A substantial amount of cross-over material, applicable to more than one computer but not necessarily to all three, befuddled our editors more than once in the preparation of this issue. We believe the end product to be as clear-cut as possible. I would encourage you to read the material in all of the sections, whether specifically geared to your computer or not. One of *SoftSide's* major goals is to encourage cooperation and sharing of information among all computerists. We speak a common language with system-specific dialects. To allow those dialects to become walls blocking communication would be a great disservice to our common goals.

Last month I spoke of the survey which appears in this issue. I reiterate that this is the most important survey we've ever published. There is no doubt that the time involved in completing four pages of questions is an investment on your part. However, it will be an investment well made for you and us as it will allow us to bring you more material geared to your needs and capabilities. Please don't hesitate to include a separate note containing your observations and recommendations concerning *SoftSide*.

We are experiencing our usual lull in submissions which falls shortly after the holiday season. This year, however, there seems to be a particular slacking off in our Apple submissions. Apple programmers take note, we have a great need for new programs so this


would be a good time to submit programs you'd like to have receive special attention from our submissions department.

Another special issue is in the making here in the offices of *SoftSide*. As plans now stand, the majority of our April issue will be devoted to word processors. We will be including a complete, updated listing of *Microtext*, the *SoftSide* BASIC word processor by Jon Voskuil. We're receiving a great deal of your documentation and article submissions on *Microtext* and couldn't be more pleased with the way you're utilizing the program. We've also been getting a number of enhancements for *Microtext* and will compile all of them in the complete listing. In addition to *Microtext*, we'll be giving you an expanded review section, evaluating the most current versions of popular word processors for all three systems.

In preliminary research for the issue, we've discovered that there is a new generation of word processors arriving for all of the systems. The revisions on those currently on the market are reportedly so substantial that several of the manufacturers we contacted refused to send review copies until the new versions were available. As I've perused the manuals, I can see why. Hold on to your hats and get ready for major changes in much of the software we've been considering "state-of-the-art."

The spring computer show season is under way and again, *SoftSide* will be present at most of the extravaganzas. Please drop by and see us. We're always happy to see some of you face to face — it makes our communication through paper, tape and disk much, much more human.

Until next month, Happy Hacking!

Randal L. Kottwitz
Associate Publisher/Editor 
SoftSide

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



by Allen L. Wold and Fred D'Ignazio

LIGHT SABERS AND LASERS

A few weeks ago one of us (Allen) attended a science fiction convention in Virginia Beach. While doing some after-hours browsing through the huckster's room, he saw a new role-playing game for sale, called *Killer*. It is a set of rules for real-life role playing, in which the players assume the roles of assassins, who try to kill each other using harmless weapons such as water pistols and rubber hand grenades. A number of convention attendees who had purchased it were playing it in the halls when not attending panels and movies.

We have been to other conventions where some form of real-life *Dungeons and Dragons* was attempted. Someone hides a treasure somewhere in the hotel (the hotel employees really love this), and the players spend hours tracking it down, with simulated combats whenever appropriate.

In *Killer*, however, there is no dungeon master. Instead, two to about a dozen players decide in advance what the limits and objectives of the game will be: from a one-on-one assassination attempt to be concluded by a specific time in one day, to a week-long campaign involving several people. The game does not require constant attention: It can be incorporated into and be worked around a day's regular activities.

Whatever one may think about the violence content of the game, those who were playing seemed to be enjoying themselves. Not a gang fight but a test of tracking and ambushing skills, it is more like training for a career as a process server delivering summonses, than as a warrior.

Though our knowledge of the game is slight, we can still see that it is a sign of some kind. Role-playing games are coming off the mapboard and out of the notebook, and getting into the real world. The point is that people want to participate more than the "traditional" role-playing games allow, to assume the role of their hero in a broader context. A psychology text could be written on the desire of people to live out fantasy adventures in the flesh. We're not going to do that.

Instead, we are going to take it as a

given that people **do** want to assume the role of a hero in more than just a paper and pencil game. They want to feel the tension and excitement of a real hunt for treasure or an enemy, but at the same time they do not want it to be real. They do not want to kill, cheat, or otherwise dominate people in real life. They only want to have the satisfaction of having done so in a "real world" fantasy.

The two facets of a role-playing game are equally important: role-playing, and game. Without the game aspect, there are no heroes, just psychotics. Critics of *D&D* and other similar games should bear this in mind.

The fantasy aspect allows players to experience a form of behavior in which they would not indulge in real life. Few people really want to enter a dank dungeon with monsters in it, no matter how much treasure there might be. The fantasy is a way of confronting certain fears and problems safely, without the danger inherent in a real-life situation.

Given that people want more direct experience, and at the same time the security of knowing that it's all fictitious, there is a lot that can be done.

Killer is not computerized, so we will not deal further with it. But there are two ways a computer can assist in achieving a direct but fictitious adventure experience, though not without other technology.

One is for the computer to provide the entire environment, assisted by wrap-around video, stereo sound, and other sensory input, such as described by John D. MacDonald in his classic story, *Spectator Sport*, which appeared first in *Thrilling Wonder Stories*, February, 1950, and has since been included in many anthologies.

It concerns a time traveler to the future. He is assumed by the authorities to be insane, and "cured." This has the unfortunate effect of destroying his mind. When the authorities discover that he really is a time traveler from the past, they try to make up for their mistake by giving him a free life-time admission to one of the dream machines, which everybody else in that time saves for all their lives to buy into.

One is placed in the machine; eyes, ears, voice, hands, and feet are connected to electronic sensing devices; and the machine presents adventures for the person to experience: from harems to old west, from soap opera to space opera. Read the story, and see one idea of the ultimate "Computer Assisted Role-Playing Game." We will be discussing this aspect of fantasy gaming at a later time.

Another way the computer can assist in achieving an adventure experience is for the computer, in some of its various incarnations, to assist in real-time playing. That is, instead of the players entering the machine, the machines are attached to the players. Which brings us to the subject of this column: exploring two facets of one way such a thing might be done.

The idea for this column came when we were discussing some of the ways computers can assist in our recreation. Suddenly a light flashed, almost literally, in Allen's mind, and he said, "computerized fencing."

There already is electric fencing. The combatants wear special jackets embedded with a light metal mesh and hooked by long wires to boxes, one for each fencer. Their foils have spring-loaded buttons at the tip, and are also connected, by another wire, to their boxes.

When the tip of the foil is depressed, it closes a circuit. If the foil is touching the floor, or some part of the opponent not covered by the mesh jacket, a red light goes on, indicating a foul. If it is depressed while in contact with the opponent's jacket, a different circuit is closed, and a green light goes on, indicating a fair hit. This reduces the number of judges necessary at a fencing match from five to one. It also removes any doubt as to whether the touch was actually made. A light graze won't depress the button. A hit on the opponent's sleeve flashes red.

Suppose, instead of a metal foil, a very weak laser were used (such as those in grocery store check-out registers), focused at a point where the tip of the foil would be. The opponent's jacket would be photoelectrically sensitive to laser light of a cer-

tain intensity, and the focus would be strong enough to signal a fair hit.

So far so good, but no computers used, and no score if you ran your opponent through — the light nearer the guard, being out of focus, would also be too faint to register.

The next logical step is the laser pistol, again using a beam of light too faint to cause any harm. The opponents would wear a mask which would completely filter out the color of the laser light, and so protect the eyes from any chance of damage. Otherwise, the laser would hurt no more than a flashlight.

Or you could have fast-draw gunfights, a la old west. Each pistol would be a laser (with sound effects if you wish), and when the fighters draw and fire, the laser beam would strike whatever it is aimed at. The players' clothes would be sensitive to the laser light, and record who hit whom where and when.

There are two ideas here: one is fencing, the other is gunfighting. Let's see where fencing takes us.

There is a toy on the market called *Boffers*, foam broadswords which allow two combatants to hit each other as hard as they like without any damage whatsoever.

You can fence with a foil, because it is flexible and has a button point, even though it is made of steel. But try fencing with a four-pound broadsword, and even with dull edges and padding, people are going to get hurt. The Japanese use bamboo swords for their fencing, which are light and flexible, but they can still give an awful whack and require lots of padding if you're not going to get bruised. So *Boffers* are the perfect solution, though they don't have the heft or weight of a real sword.

The Society for Creative Anachronism fences with real swords (blunted) and in armor. But not everybody has access to the SCA, nor has the time the SCA demands of its members to fully develop and learn the skills necessary before they are allowed to fight. Again, the *Boffers* are a solution. But let's take them a step further.

Suppose the lightweight plastic sword were loaded with strain gauges. When you strike an opponent, the gauges tell how hard you hit. Calibrated to a real sword, the gauges would tell you whether you just nicked the fellow, or cut him through. When the two swords hit each other, both record a hit, but contact between the two negates the score. Only contact on the opponent's armor (perhaps lightweight Mylar with circuitry

embedded as in electric fencing) would allow the computer-controlled gauges to score a real hit.

You could do the same with battle axes, morning stars, maces, or any medieval weapon. Wired to boxes controlled by a computer, you could fight away to your heart's content without any fear of damage. Only contacts on the opponent would count. The weapons would transfer to the boxes the amount of force applied, the location of the hit, and so on, and disabling hits would be signalled and recorded. No more shouting: "I hit you first." "No, you hit my shield." "I did not. I hit your shoulder."

The computer would know, by the nature of the circuit closed, just who hit whom where and how hard, and keep track of all hits. As players developed, their "constitutions" would improve, and they would be able to take more hits: *D&D* combat in the flesh, with all the character promotions included. You assume a role, the parameters are entered into the scorekeeping computer, and you fight with the handicap of your character.

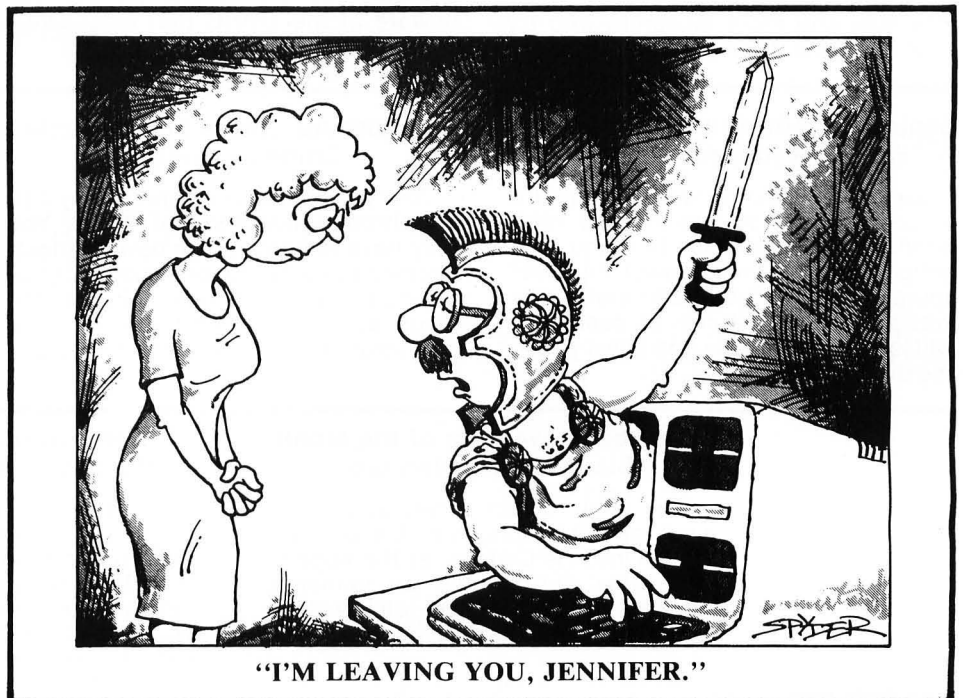
Let's take this into the future, a la *Traveller*. Instead of swords, we have blasters, rifles, lasers, and so on. All the simulated weapons would actually be low-powered lasers, each calibrated a different way to simulate different effects. Costumes would incorporate a sensing mesh, which would detect when and where a beam hit. The computer scorekeeper would have to be no bigger than a paperback book and could be worn on the helmet or in the breast pocket.

Let's assume, for the sake of simplicity, only three classes of weapons: the lightweight laser, the pistol or rifle, and the blaster. Similarly, assume only three classes of armor: the bullet-proof vest, the light-jacket, and the heavy armor.

Lasers penetrate vests well, armor a little, and light-jackets not at all. Bullets are stopped by the vest, can damage the armor, and penetrate the light-jacket easily. Blasters take out armor, and are reduced by the light-jacket or vest. Lasers are concealable, pistols and rifles are not, and the blaster is expensive. Vests are cheap, light-jackets concealable, armor is prohibitively expensive and obvious. Your choice of weapons and armor is determined by your character and his or her resources.

The computer in your costume not only knows when you've made a hit, but by what kind of weapon, and, by the intensity of the light, at what range. The computer would keep score of glancing blows, direct hits, and so on, and tell the player how much damage he or she has received, and how that would limit future actions. The computer would also keep track of recovery.

For role-playing in real life, the combination of computer and laser technologies offers many opportunities. Right now the equipment might be too expensive, and probably doesn't exist yet, but it will soon. And before too long, watch for computerized game costumes and weapons sold at Radio Shack for the price of a handheld electronic game or calculator. ♪



"I'M LEAVING YOU, JENNIFER."

Your Adventures



June Adventure of the Month Arabian Adventure

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

July Adventure of the Month Alien Adventure

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

August Adventure of the Month Treasure Island Adventure

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

September Adventure of the Month Jack The Ripper Adventure

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

October Adventure of the Month Crime Adventure

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

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

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

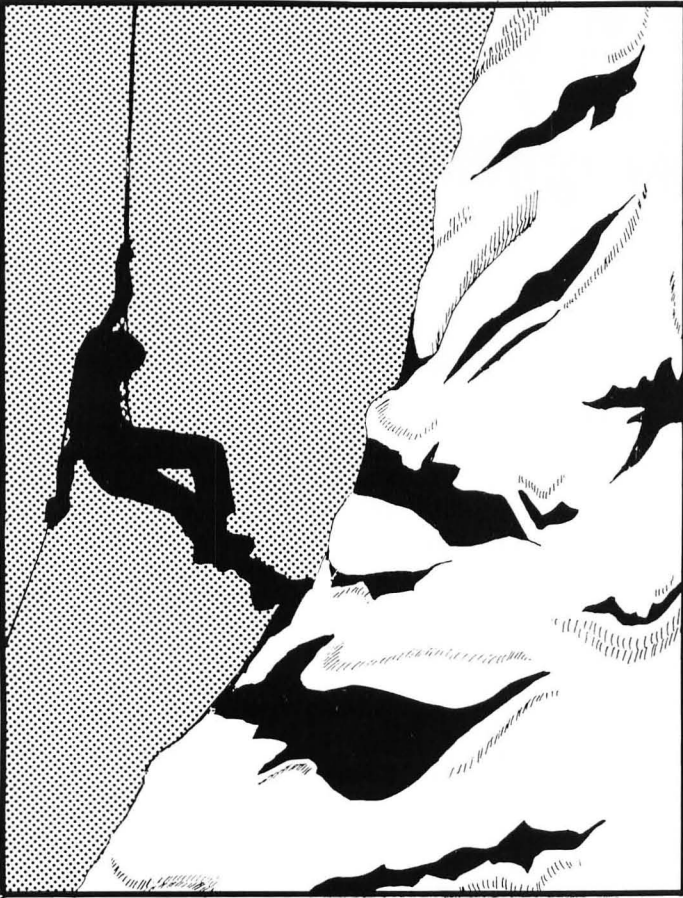
December Adventure of the Month Black Hole Adventure

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

January Adventure of the Month Windsloe Mansion Adventure

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

will start here —



FEBRUARY ADVENTURE OF THE MONTH: KLONDIKE ADVENTURE

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

Subscribe to Adventure of the Month

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

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

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

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

Adventure of the Month 6 South Street, Milford NH 03055

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

■ **Six month subscription:**

- Cassette (\$27)
- Disk (\$45)

■ **Individual adventures (please specify)**

- Cassette — \$6 each
- Disk — \$9 each

■ **Three adventures on one super disk (\$24 each):**

- Arabian, Alien, & Treasure Island Adventures
- Jack the Ripper, Crime & Around the World Adventures

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 _____

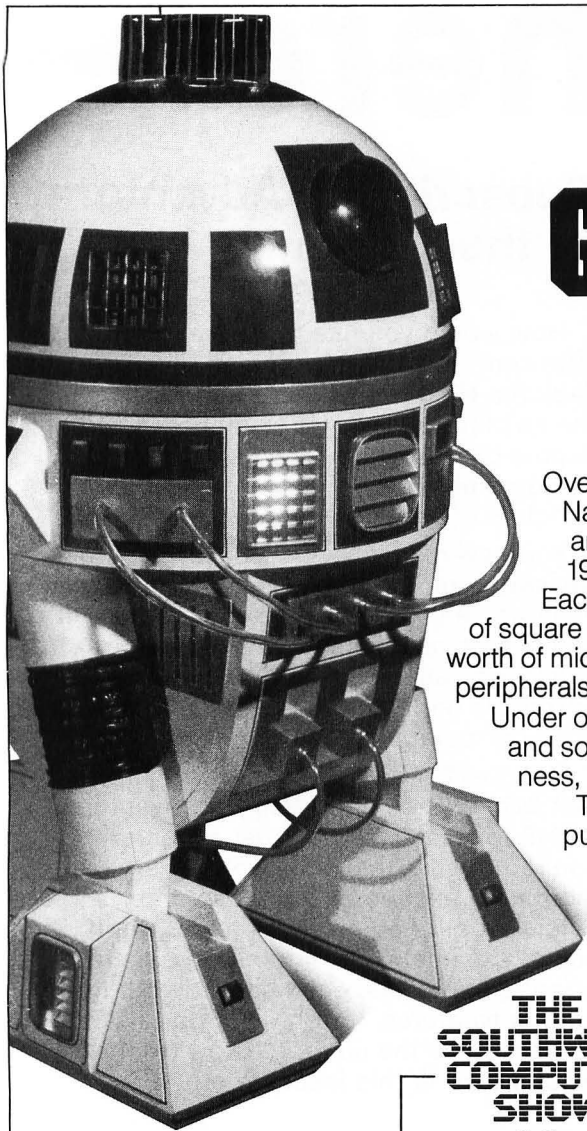
Payment enclosed

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.



HAVE WE GOT A PROGRAM FOR YOU IN '82

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

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

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

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

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

THE NATIONAL COMPUTER SHOWS AND OFFICE EQUIPMENT EXPOSITIONS

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 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.



A Perfect 10

by "J"

Computers, in contrast to women, are logical. If you think that's a sexist remark, just ask any man — he'll tell you it's true. Actually, it's just as much a "machinist" remark, don't you think? And speaking of logic, what about the English language? If a machinist is a person who is skilled in the use of machines, then a sexist . . . well, never mind.

Anyway, as all faithful *Star Trek* watchers know, computers are inexorably logical. All you have to do is to confront them with some good old human illogic, and they self-destruct. If it weren't for this fact, Kirk and Spock would not have come back alive from several of their missions. Logical convolutions, it seems, are sometimes mightier than the phaser.

A nineteenth-century mathematician named George Boole is the father of mathematical logic as used by computers today. (You'll occasionally hear the adjective "Boolean" attached to things having to do with logic and mathematics.) Without George, things would be very different. The "New Math" might never have come along to aggravate the generation gap between parents and their school-aged children. And you might still be able to hear a real, live human being tell you that your order is ready for pickup at the Sears catalog desk.

Logic is a big subject, but at the moment I'm concerned only with the use of what are called "logical operators" in BASIC programming. Used in conjunction with other animals called "relational operators," these enable you to do some fun and sometimes



tricky programming with ease.

Logic is concerned with truth and falsehood. Consider the following statements:

- (A) 3 is equal to 3.
- (B) 2 plus 8 equals 9.
- (C) X is equal to Y.

It's quite obvious that statement A is true, statement B is false, and statement C might be either true or false, depending on the values of X and Y. You can look at any such statement and mark it with a T or F just as you do on true/false tests. Or, if you were a computer (or a mathematician, which is pretty much the same thing), you might prefer that truth and falsehood be marked using 1 and 0 instead. Then statement A has a logical value of 1 (true), statement B a logical value of 0 (false), and statement C an unknown logical value (but either 0 or 1).

Now, if A has a logical value of 1, then the opposite or negation of A must have a logical value of 0. The logical opposite of statement A is

- (A) 3 is not equal to 3

which is indeed false, and therefore has a logical value of 0. Another way of saying this is that if A is true, then NOT A is false. Likewise, since B is false (0), then NOT B must be true (1). True??

If you find this at all confusing, you'd better go get a pizza or something before continuing.

It's possible, then, to combine such simple logical statements together, to create a longer logical statement. And this longer logical statement, like each of its components, must be either true or false. The logical operators used to combine simpler statements into more complex ones are AND and OR. Let's try combining statements A and B in two different ways (at the same time, I'll shorten the verbiage by using the algebraic operator + and the relational operator =):

- (D) $(3 = 3)$ AND $(2 + 8 = 9)$
- (E) $(3 = 3)$ OR $(2 + 8 = 9)$

How do we evaluate the truth of these statements? Well, if two statements are joined together by the logical operator AND, then both of the statements must be true in order to make the overall statement true. That's logical enough. It could be phrased this way: "Is it true both that $3 = 3$ AND that $2 + 8 = 9$?" It's like those occasional tricky true/false questions that your teachers used to give you, where you had to make sure that EVERY part

of the statement was absolutely correct before you could mark it true. Like, "The capitol of the United States is Washington, D.C." Is he trying to trick you by spelling it "capitol" (the building) rather than "capital" (the city)? Or is it just a typo? Or doesn't he know the difference himself?? Strictly speaking, the statement is false, because one of its parts is false.

That's the way it is with a logical AND: If any part is false, the whole statement is false. Therefore statement D is false, because statement B is false. In terms of 1s and 0s, this means that all the statements strung together with ANDs must have a value of 1 in order for the overall statement to have a value of 1. A single 0 in the sub-statements will blow the truth of the

■ You can use ORs to connect a series of 16 false statements and one lonely true statement, and the overall statement will be true because of that one true element. ■

whole thing.

The logical OR is quite different. Statement E can be phrased this way: "Is it true either that $3 = 3$ OR that $2 + 8 = 9$?" Well, sure. One of those things is true; therefore the whole statement is true. You can use ORs to connect a series of 16 false statements and one lonely true statement, and the overall statement will be true because of that one true element. Every last one of the statements must be false (0) in order to make the whole thing false (0).

Things can get a lot more complicated when you start mixing ANDs, ORs, and NOTs together in one long string of statements. And, of course, you're not limited to working with just = as a relational operator. Also available are < (less than), > (greater

than), and the combinations \leq (less than or equal to), \geq (greater than or equal to), and \neq (less than or greater than; i.e., not equal to). But no matter how complex the statement, it can be taken apart step by step, each substatement being evaluated separately and then combined into larger and larger pieces, until the truth of the whole statement is determined.

Just for the fun of it, consider the following statement:

- (F) $((8/3 > 3)$ AND (1 meter = 100 centimeters)) OR (NOT (helium is the lightest element) AND (a square has four sides) AND $(N = M)$ OR NOT $(N = M))$)

It looks a bit confusing at first, but let's break it down into logical units and assign them values:

- $(8/3 > 3)$ is false (0)
- (1 meter = 100 centimeters) is true (1)
- (helium is the lightest element) is false (0)
- (a square has four sides) is true (1)

The statement $(N = M)$ might be either true or false; but in the context of the overall statement, it doesn't really matter. If true, then NOT $(N = M)$ is false; and if false, then NOT $(N = M)$ is true. Either way, we get the same result for the composite statement:

- $((N = M)$ OR NOT $(N = M))$ must be true (1)

The whole statement, then, can be reduced to this, in a kind of shorthand notation:

- $(0$ AND $1)$ OR $($ NOT 0 AND 1 AND $1)$

That looks a little more manageable. Now, in order for the whole statement to be true, either one (or both) of the statements in parentheses must be true. Looking at the first, $(0$ AND $1)$, it should be evident that this evaluates as 0 (false), since not all the elements are true. Looking at the second, $($ NOT 0 AND 1 AND $1)$, we find that it does evaluate as true, since all three elements are true. (Remember that NOT 0 is the same as 1.) This being the case, we arrive at the logical conclusion that statement F really is true. Just what you always wanted to know, right?

Computers are awfully good at this sort of logical analysis, as you might expect. And they understand the words AND, OR, and NOT, just as I've been using them. So finally we get to the application of all this to programming in BASIC.

As you know if you do any programming at all, one of the most common aspects of programming is giving the computer instructions based on some kind of decision-making process. If a certain condition is met, then you want to have the program do one thing; if not, then do a different thing. The usual way of writing such conditional instructions into programs is to use IF...THEN statements. For example:

```
100 INPUT N
110 IF N > 0 THEN PRINT
    "POSITIVE"
120 IE N < 0 THEN PRINT
    "NEGATIVE"
130 IF N = 0 THEN PRINT "ZERO"
```

When you give the computer instructions such as these, what you're really asking it to do is to evaluate the truth of the statements ($N > 0$), ($N < 0$), and ($N = 0$). If it comes up with a logical true value, it executes the instruction; if a logical false value, then it doesn't.

To illustrate this, let's rewrite these lines in an unconventional (but perfectly legal) form:

```
200 INPUT N
210 A = (N > 0)
220 B = (N < 0)
230 C = (N = 0)
240 IF A THEN PRINT "POSITIVE"
250 IF B THEN PRINT
    "NEGATIVE"
260 IF C THEN PRINT "ZERO"
```

The variables A, B, and C are used to hold the logical values of the three statements in question. In this example, one and only one of these must be true. So for any number that you might enter, the computer will assign a false value (0) to two of the variables, and a true value to the third. (The TRS-80® and some other computers use the value -1 to represent a logical true value. The Apple and ATARI® both use +1.) This logical value is then used in the appropriate IF statement: The computer looks at the numerical value of the variable and, in the IF...THEN context, interprets that as a logical value.

This second set of lines, then, will function in exactly the same way as the previous set, with the added feature that the logical truth or falsehood of each of the three statements is stored in memory for future reference. Such variables, which signal the truth or falsehood of some condition, are often

called "flags." Lines 240-260, by the way, could also be written in the form

```
240 IF A = 1 THEN PRINT
    "POSITIVE"
```

but the "= 1" (or "= -1") is quite unnecessary.

It's certainly interesting, and occasionally even useful, to set up an IF...THEN line in the way we've just done. But some of the more practical applications of logical statements in BASIC programming are designed precisely to avoid having to use the IF...THEN structure. For example, let's say that you want to print a column of numbers so that they are properly aligned under one another. Let's say you know that they'll all have

```
550 PRINT TAB(T);N
```

(Line 550 would need to be revised for ATARI® BASIC.)


Using logical statements apart from the IF...THEN structure, however, this coding can be simplified somewhat, and speeded up as well:

```
600 T = 10
610 PRINT TAB(10 + (N < 1000) +
    (N < 100) + (N < 10) + (N < 1)); N
```

(All the plusses in line 610 would need to be minuses on the TRS-80®.) Each of the logical statements in parentheses is evaluated for truth (1 or -1) or falsehood (0), and the resulting numerical values are simply added to the tabbing location, producing exactly the same effect as the previous set of lines.

This is a useful kind of technique to use in creating densely packed code with as few lines as possible — for instance, in writing One Liners. If you are not forced to use the IF...THEN structure for decision-making, then you can pack a lot more sophistication into a single line than would otherwise be possible (especially if you don't have an ELSE statement to work with).

There are also occasions for wanting the computer to evaluate the truth of some group of conditions, such as those in statement F, above. I recently wrote a program to analyze the responses on a survey form. The information I wanted from the survey was of the type, "How many people who own Apples also own modems?", or "How many people who own either a Model I or a Model III also own at least one disk drive and a printer?" Obviously this was a natural for using logical operators to do the analysis. The program looks at each survey in turn, checking to see if there is a positive response to each of the questions in question. It then evaluates the group of responses using the logical operators AND, OR, and NOT, and comes up with a logical 1 or 0 for that group of responses for that individual survey. After scanning all the individual surveys, it then has a tally of how many logical 1s it found, which tells me what I want to know.

A creative programmer will find it fun to play with logical statements and operators in his programs, finding ways to use them that make the flow of the program clearer and save time and memory. If you haven't used these animals much, you'll want to experiment with them and get used to the ways in which they work. I hope that the few applications I've suggested will stimulate your inventiveness. 

☞ If you are not forced to use the IF...THEN structure for decision-making, then you can pack a lot more sophistication into a single line...☞

values between 0 and 9999. The easiest way to do that would be with a PRINT USING statement such as the TRS-80® has. But what if you don't have such a statement available? You can then TAB to the appropriate place on the line, or PRINT an appropriate number of spaces, so that the number starts printing in the correct column for its number of digits. This obviously involves testing to see how large the number is. The following lines would do the trick (where N is the number to be printed):

```
500 T = 10
510 IF N < 1000 THEN T = T + 1
520 IF N < 100 THEN T = T + 1
530 IF N < 10 THEN T = T + 1
540 IF N < 1 THEN T = T + 1
```



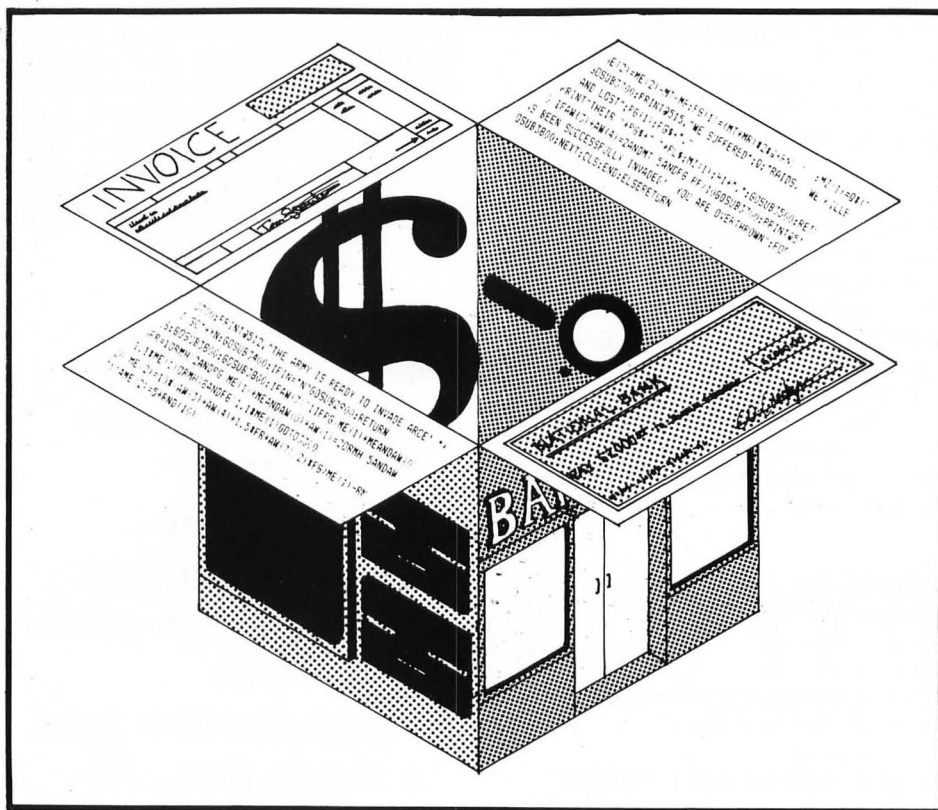
by Lance Micklus

GETTING A BIT SERIOUS — PART 7

In the December *SoftSide*, I told you about Kim Foshier, my office manager, and her difficulties in learning how to run a Model II *General Ledger* program and when to press the ENTER key. I'm happy to report that Kim's really getting into this computer stuff now. During her lunch hours, she's been playing *Space Assault* on the Color Computer. Her high score is 37,430 compared to my high score of 11,800. It seems to me that if I could hook a joystick up to the Model II and work that feature into the *General Ledger* program, all my bookkeeping problems would be solved.

The saga of *The Mean Craps Machine* has finally been resolved. Scott Adams called one afternoon and said he had looked at the game and liked it. The Color Computer version looked like it had a lot of market potential but he didn't think the Model I/III version would do as well. Frankly, I had to agree with Scott. A couple of years ago, *The Mean Craps Machine* would have been an excellent product. But now, the Model I/III market is too competitive. So, Adventure International is now selling it for the Color Computer. I'm selling it myself for the Model I/III. Later, when the time is right, it may show up in another form. Perhaps there might be a *The Best of Lance Micklus* disk and it will be on that.

The Color Computer *Star Trek III* game is also available from Adventure International for 32K machines with a 16K version possibly ready by the time you read this. It is an interesting story. A few advance copies were sent to some loyal friends. One of these friends converted his Color Computer to 32K just to run the game. We had put blinking dots in some of the displays as an aid to aiming via the joysticks. Guess what? My test game player with a Color Computer is colorblind and he couldn't see the dots. Back in the days when I worked in television, I remember that we always checked our TV graphics on a black and white monitor to make sure they held up. It just never occurred to me that somebody with a Color Computer



would not see things in color for whatever reason. So I checked it out and found that in black and white, the dots barely show up.

I felt that this was much too special a case so I didn't bother to change it. Then Scott Adams called back and complained that many of the joysticks for the Color Computer are flaky. We gave up. We added a digital entry mode just like the Model I/III versions for those with black and white TVs, colorblind people, or those who have flaky joysticks.

Much to my surprise, the Color Computer market has really been buying my products. This market is turning out to be much larger than I had thought. At one point, I was also seriously thinking of getting into the ATARI® market with a stronger product line than I have now. But with all of this controversy about ATARI®'s copyrights, I think I'll wait for something else to come along.

This month marks the last install-

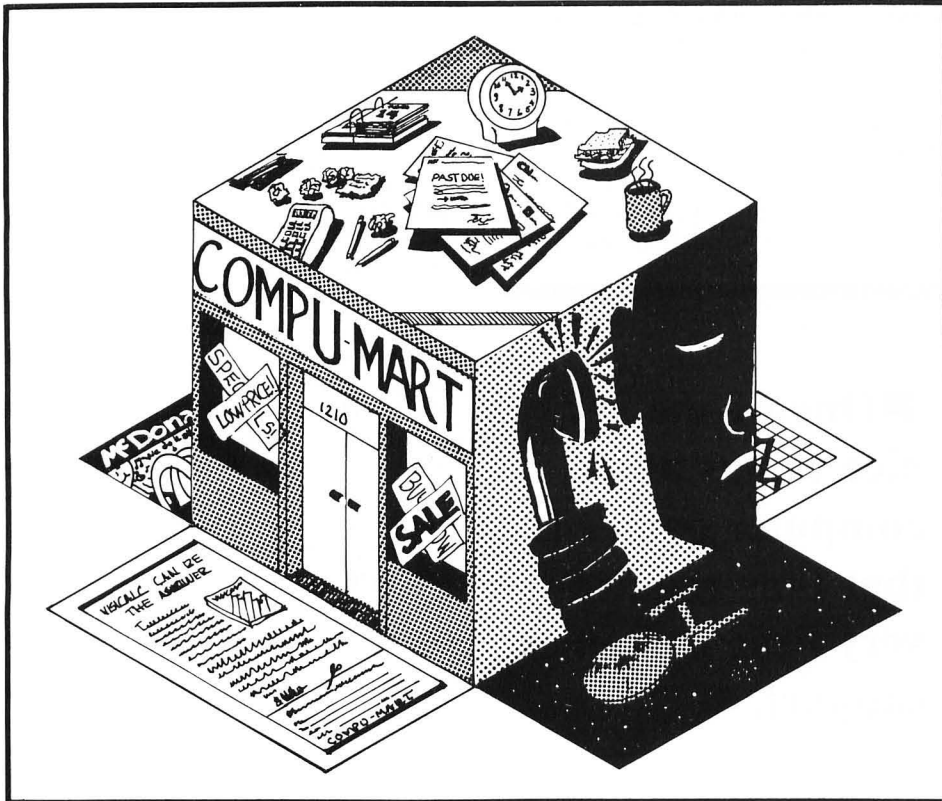
ment of the GETTING A BIT SERIOUS articles. Once again, we start off with my favorite subject — money.

Figuring the Right Price

One of the most important things to be considered when selling anything is to price it right. An item can be priced either too low or too high and end up getting killed because of it.

Actually, there are two prices to consider. The first price is the minimum price you must get to make enough money to justify making the product in the first place. The second price is based on the value of that product to the consumer and it can vary greatly from one person to the next. We will call these two prices the seller's price and the buyer's price.

When the seller's price is below the buyer's price, you have an ideal situation. Assuming some middle ground is found, then the seller gets more money



than he really needed to get and the buyer feels he's getting a deal because it's worth more than it's costing him. If the two prices are the same, a transaction can still take place but both parties aren't as happy about it. The worst situation, of course, is when the seller's price is higher than the buyer's price. Then usually no transaction can occur.

I remember when *Visicalc* first came out, somebody on MicroNet left a message complaining about the price of \$99. Basically, what this person said was that he didn't think it was worth \$99, but if Radio Shack would have priced it at \$25, then they would sell a lot more *Visicalc* programs and he would have bought one. What this person was really saying was that *Visicalc* was worth \$25 to him. I know one person who got a \$250,000 loan from a bank based on his *Visicalc* projection sheets. Even though this was a small company, the fact that they had the ability to produce this type of projection convinced the bank that this com-

pany would be a good loan risk. To this company, *Visicalc* was worth a lot more than \$99. In fact, I've used *Visicalc* myself in exactly the same way and talked my bank into setting me up with a very nice line of credit. Although I might have gotten the loan for my company without *Visicalc*, I'm certain it would have been a lot harder.

Let's consider this further. At \$99 a pop, *Visicalc* is going to at least be equal to if not much less than the buyer's price for a lot of people. For others, such as the fellow who left the message on MicroNet, it is greater than his buyer's price. Just to keep it simple, let's assume that it doesn't cost Radio Shack a cent to make *Visicalc* so the selling price is pure profit. The question now becomes this: If *Visicalc* sold for \$25, would Radio Shack sell at least four times as many *Visicalc* programs as they do for about \$100? Think about the people you know who don't own *Visicalc*. Would they buy it for \$25? Really? Some of them would, but

I think there would still be a lot of people who wouldn't bother because they just have no use for it even at \$25. What happens if *Visicalc* sells for \$10? Now, I think we've come to the point where anybody who could possibly find a use for *Visicalc* would buy it, but would that increase the sales by a factor of at least 10?

There is no definite answer to these questions; you have your opinion and I have mine. I think that the minimum price Radio Shack could sell *Visicalc* is \$50. I doubt, however, that if they did cut the price in half, they would double their sales. On the other hand, if Radio Shack sold *Visicalc* for \$10, then just about everybody would buy it and their sales would increase by at least a factor of 10. There's just one problem with that — they would lose money on every sale. So the more they sold, the more money they would lose. It would be better if they didn't sell it at all.

There is a name for all of this: It's called marketing projections. It is an art, not a science, because nobody really knows the answers for sure. By studying your market carefully, you can make some pretty good guesses and be right most of the time. Let me ask you something. Have you ever seen a McDonald's restaurant that went out of business? I don't think there's any such thing and there are two reasons for it. First, McDonald's has been at this hamburger thing long enough to know exactly where to pick a good location. They don't put a McDonald's in unless they know from their marketing studies that the location is correct. Second, they require so much money to get a franchise that the only ones who can afford to build a McDonald's have already proven themselves to be good businesspeople. In other words, it weeds out the losers.

McDonald's is a prime example of success building upon success. No McDonald's restaurant has ever failed. So businesspeople find this to be a very secure investment, which makes more McDonald's, which become profitable and make more people want to buy into the franchise.

It's also the secret to Radio Shack's

success. Their marketing people seem to be on target almost all of the time. That's why many of the fringe businesses, like mine, prefer to support Radio Shack products over all others. If the XYZ company comes out with a new computer, maybe it will bomb. If it is a Radio Shack computer, chances are it will sell well because TRS-80® computers have always sold well before. When gambling, the wisest thing to do is to play the best odds.

Obviously, another factor which effects the price is the cost of making the product. There are costs and there are costs. The easiest ones to nail down are the out-of-pocket expenses for such things as disks, instruction manuals, and so on; but there are other costs which must be considered.

Some of these hidden costs are for advertising and general overhead. But the big killer is customer support. One of the nice things about computer games is that they require very little customer support. When you get into business software things change in a hurry. Many business customers know very little about computers or programming and you really have to hold their hands. At least with *Star Fighters*, everybody plays by the same rules. Payroll programs must be written to play by 50 different rules — one set of rules for each state. I think you can see what the problem is.

The third factor is the development cost. This can be major or minor depending on the quantities we're talking about. I remember when I was in television, we could see the effect of that on the price of equipment. A few years ago, our station went out to bid on the cost of three studio-quality color television cameras. RCA won the bid with a camera called the TK-45A. The price of each camera (without lens) was around \$90,000. The ones we received were serial number 1100. That tells you roughly how many of the TK-45 series RCA had made. One of the reasons they were able to bid aggressively was that the camera was selling better than they expected, so they had already made back their development costs. This put them in a better position to discount the price in their bid. Our timing on the bid was very good. About a year later they switched over to the TK-46, fixing a number of problems in the TK-45. Last I heard, they are now on the TK-47A, which is computerized. You just push one button and the camera sets itself up while the technician watches.

The question we started with was how to figure out the right price. The way I do it is to try to work backwards. First, I try to determine how many I

can sell. Then, I try to figure what the seller's price should be. Now, I can work backward to figure out what my cut will be and I know how much money to invest in the project. This works well if the seller's price is relatively inelastic, which it usually is for games.

Sometimes you have a product where the seller's price is very elastic. *ST80-III* is one of those products that could sell anywhere from \$50 to \$600. In this case, I try to target *ST80-III* for a particular group. There are two: the serious hobbyist who wants a top-of-the-line product and the commercial user who will use *ST80-III* for some business application. To simplify this, we will talk only about the Model I and Model III versions. Reducing the price would generate more hobbyist sales but would not increase the sales to commercial users. Since half of the *ST80-*

■ One of the very nice things about computer games is that they require very little customer support. ■

III programs are sold to commercial users, if I cut the price in half, then four times as many hobbyists would have to buy *ST80-III* than do now, just to break even. That's not going to happen. On the other hand, if I double the price, it would kill my hobbyist market and would reduce my commercial sales somewhat. I'd end up making less money at \$300 because my sales would cut in half. But, if I increased the price to \$600, my sales to commercial users would drop to 25% of what it is now, so I'm doing no better than I was at \$150. Of course, it's all theory but it's interesting to think that *ST80-III* would make just as much money at \$150 as it would at \$600, but would lose money at any other price.

Given the above choices and the chance to do it all over again, I'd still sell *ST80-III* at \$150 because at that price, I get one thing I don't get when I

sell it at \$600 — I get more famous because more people will have the thing I made. I never told you I was modest; besides, if you can't be rich, at least be famously poor.

Seriously, though, there is another reason for the \$150 price: Retail people want some spread. They want a cheap and dirty terminal program, a mid-priced one, and a top-of-the-line one. Nobody wants to sell a top-of-the-line terminal program at half the price. If the customer wants an \$80 terminal program, then retailers want to sell him an \$80 terminal program.

In case you think that sounds obvious, read the ads in the computer magazines. What you'll find are ads for \$150 terminal programs (because they're supposedly as good as *ST80-III*) that the competition is selling for \$80. OK. But look further and you'll find that the competitive products are almost always sold by one company — usually the company which is producing the product. Nobody else wants to retail it; now you know why. But, please, don't stop at just terminal programs. Have yourself a field day and look at the ads for business software. It seems as though some of the people who produce business software don't know much about business.

Pricing, then, becomes a matter of first sizing up the market. You have to determine who your potential customers are, how many customers there are, and what they are willing to spend to obtain the thing you're planning to make. That gives you the gross sales figures in dollars from which you can work backward to figure out how much money you can afford to invest in the project and whether or not it will be profitable. It might turn out that some idea of yours isn't financially feasible because your cost is so high that there just aren't enough people willing to spend that kind of money.

To be very honest with you, though, I tend to work in a different way. More often than not, I tend to get some crazy idea for a program and then I have to write it. So I come at the whole thing from the other direction and my problem is how to make the thing pay for itself. So I figure out how much the program should sell for and tweak it up until the program is good enough to demand the price I need. Of course, sometimes things don't work out and you end up with a pet program that doesn't make any money.

Customer Support

One of the things that you have to include as part of your cost is the overhead for customer support.

No matter what product you sell, it is going to be used by all kinds of people. Most people are pretty good about trying to figure it out first. Only when they've tried everything they can think of, do they call or write. Whenever I hear from someone like that, I always feel partly to blame for not making my instructions clearer.

However, I quickly run out of patience with people who do not read instructions. I cannot "learn" you something. You must learn for yourself. I can only teach. This is why a lot of companies do not have WATS lines for customer service. It's too easy for people to pick up the phone and expect someone on the other end to tell them what they could have read in the instruction manual. Companies that do have WATS lines work around that problem very nicely. They never have enough lines to handle the traffic. You have to work at it to get through. This forces people to consider their questions carefully before they dial.

I recall one phone call I got from a guy who bought my *Star Trek* program. He said he couldn't get the tape to load and wanted to know what to do. In what must have appeared as a stroke of genius to him, I suggested that he return the tape to Adventure International with a note explaining that the tape wouldn't load.

Some people expect too much. I recall one guy who called and got my answering service. He demanded that his call be returned immediately even if they had to call me at home. The answering service took his number and did call me at home in the middle of my supper. Being wise, I finished eating first and then returned the call as soon as I could. It turned out this gentleman had bought my *Star Trek* game also. He got the tape to load but one section of the program was bad. He wanted me to read to him over the phone what the defective lines should be — about 30 lines of the program. I explained to him that I was at home and did not have a copy of the program with me and that, furthermore, it would take a lot of time to read 30 lines of that program. And even if I did, the chances are he still wouldn't get all 30 lines perfect. Then I asked him if he was able to load the second copy of the program. It suddenly surprised him to discover that the word DUPLICATE on the other side of the tape was not the name of a program.

Whenever I write instructions, I always assume a certain amount of knowledge on the part of the user. Over the years, the amount of knowledge I've been assuming is slowly approaching zero.

I've had one customer who wanted his money back on *ST80-III* because it didn't work at all. He said he had tried a friend's copy of *OMNITERM* and it worked perfectly. After talking with him for a few minutes, I determined that the reason his *ST80-III* didn't work on his Model I was because he didn't have an RS-232 board. (I had him remove the cover and check.) He insisted, in spite of that, that *OMNITERM* still worked! I insisted that *OMNITERM* wasn't that good.

I had a local Model III customer call asking for a terminal program to use with a DEC 2060. I told him that *ST80-UC* was all he needed and he bought it assuming it was "ALL HE NEEDED." The next day he was on the phone complaining that it didn't work. The problem turned out to be that he didn't own a modem.

Sometimes, the complaints are the

... I quickly run out of patience with people who do not read instructions.

result of a critical step in the instructions being left out. I've had people who have bought *ST80-III*, who have set up the translation table, who have set up their auto log-on, and who have gotten the program to load with no problem. They've overcome all of the difficult things and have had the "½ ½ ST80 READY ¼ ¼" on their screen. But, the program is dead — nothing happens. The reason for this apparent failure is that they didn't dial the host computer on the phone. You've got to call it before you can talk to it.

Fortunately, these types of calls are rare and they are something you have to live with. The real customer support problems come from things outside of your control. For example: You produce a program to run on the Model III under TRSDOS 1.2 and it works perfectly. Then, Radio Shack changes

the rules and now your program doesn't work on TRSDOS 1.3.

This is not to say that manufacturers are never wrong. Sometimes, they create their own problems. The point is that you have to figure the cost of customer support correctly or you're going to get hurt. One example of this type of mistake is my *Deluxe Personal Finance*. Originally, TSE sold it for \$25 on disk. Unfortunately, it required a lot of customer support. At one point, one in every three sales generated a telephone call. When SBSG got the rights to the same program, they decided to rewrite the manual and add a second disk with sample data. This new improved documentation greatly decreased the amount of customer support but replaced it with much higher production costs. Even at \$35 on disk, the program barely makes any money. As a result, SBSG isn't really trying to push it and neither am I. In fact, I really don't care if you buy it or not. Considering that I'm the author of it, that's not the kind of attitude that promotes sales. The problem is simple: The cost of customer support is killing the program. You either pay for it on the phone or you pay for it in the cost of the manual and extra disk.

My, How Time Passes

When I first started writing this series, my mouse was middle-aged. Now the mouse is dead and we have a new mouse. When I started, I had a girlfriend. Now, Dianne and I are celebrating our first anniversary. George Blank and I talked about doing this series, Dave Albert helped get me started, and now the editorial department has changed hands again. There were companies with names like Virtual Technology and The Bottom Shelf selling software — now they're gone. Certainly a lot has happened.

I really learned a lot by writing this series. That may sound strange, but it's true. It has forced me to think some things out that I had never thought about before. But the second thing I learned was far more important and I credit Dave Albert for this. If you still have copies of the earlier *SoftSides* where this series appears, I think you can see what happened.

When I started out, I took the approach that I knew and you didn't. Unless you knew what I knew, you were going to run into trouble. So, Lance was going to explain all of this to you to save your soul. When Dave got Part One of this series, he and I talked several times. Dave kept telling me to change my approach — just

share the experience. I didn't totally understand then what he wanted, but I think I understand it a lot better now.

The event that finally clarified it all to me was an article in *STARLOG*, a science fiction magazine. Every month, David Gerold, who wrote *The Trouble With Tribbles* episode for the *Star Trek* TV show, writes an article. In talking about storytelling, he stressed that a story shares the experience of the main character. This character confronts a situation that he can not deal with, but must resolve. Certain events happen which change this character. This is called learning. At the end of the story, the character armed with his new knowledge confronts the problem and overcomes it.

The movie *Close Encounters of the Third Kind* is a beautiful example of this. Roy Neary sees a UFO — a situation he is unable to deal with. His wife thinks he's nuts and his boss fires him. The image of Devil's Tower keeps driving him and won't go away. He must know if it's real. The real crisis in the movie occurs in the helicopter when he decides to take off his gas mask and risk his life by breathing what everyone says is poisoned air so that he can escape from those who are trying to prevent him from reaching Devil's Tower. Up until this moment, nothing was ever important in his life, so he never took life seriously. Now, for the first time, he is faced with something important and he get serious — he risks his life by removing his mask and discovers that the air is not poisoned. From that point on, no one thinks he is nuts. More important, events are no longer controlling him. He is now in control of the events which follow. This now makes it possible for him to know what he could never have known before — it's all real.

From a story stand point, *Close En-*

counters of the Third Kind (CE3K as it is abbreviated) is excellent storytelling.

We share the experience of a character, find out what changed his life, and then find out how that effected him. What I didn't understand when I started this series was that the storytelling technique can also be applied to other types of written works. So the change in my writing style has gone from the role of the teacher to the role of the storyteller. In this new mode, I feel that I am much more effective.

Besides myself, I know my wife has benefitted from this series. She's never completely understood what happens down here at the office. It's nice to have some understanding of exactly what it is that your husband does to make a living.


My motive, up front, was to give the average computer user a chance to see things in detail from another point of view. That's a very noble cause and I think I've accomplished that. But, if you read between the lines, you'll discover my real motive. There is a tendency for people to think that a lot of people in this business are getting rich. For example: You look at *ST80-III* and determine that 1,000 of them have been sold. Doing some simple arithmetic, you conclude that I've made \$150,000. I wish! Unfortunately, I only see a small percentage of that money — perhaps \$20,000 total over a period of three years, out of which I must pay some overhead and live on the rest. Even without the overhead, that's less than \$7,000 of income per year — hardly rich. Fortunately, there are other programs which also generate income or I would definitely be poor.

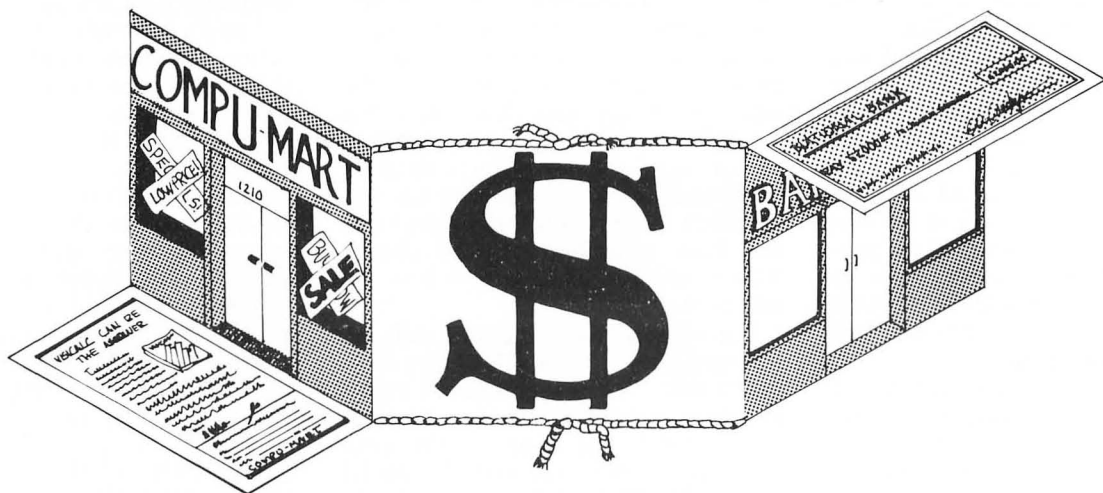
Many users in this industry are business-oriented and understand a lot of this. Many others know very little about all of this money stuff but are interested to some degree to know why a

thing which costs \$15 to make, sells for \$150.

There are a few who won't believe any of it. They figure if something costs \$15 to make, then \$30 ought to be enough. They think the development and customer support costs should come out of the profits because there will still be plenty of money left over. My reaction to this is that I'd like to see these people stop complaining about how badly they're getting ripped off and see them go into business and prove me wrong. The loudest boos always come from the people who are sitting in the free seats.

The most important thing I've tried to do for you, the reader, is to help a small minority of you decide whether or not this is the way you'd like to make your living. If you're one of those people who thinks you can sell a product that costs you \$15 for \$30, then I hope I've scared you away. If not, I fear that you may take me up on my offer to prove me wrong and find out that I'm right. Not that the principle is important, but you may end up losing your car, your house, and your life savings in the process. If I've prevented one person from doing that by writing this series, then I'd consider it worth all the time and trouble. On the other hand, I'd like to think that I might have helped somebody become better prepared for a career in this industry. Perhaps my series of articles may have helped someone who would have failed, become a success by being better equipped before starting out.

Now that all of you understand what I'm doing, and my wife understands what I'm doing, I think it's time for me to get back to doing what everybody understands that I do — at least for another two months when I again have the opportunity to fill *My Side of the Page*. 





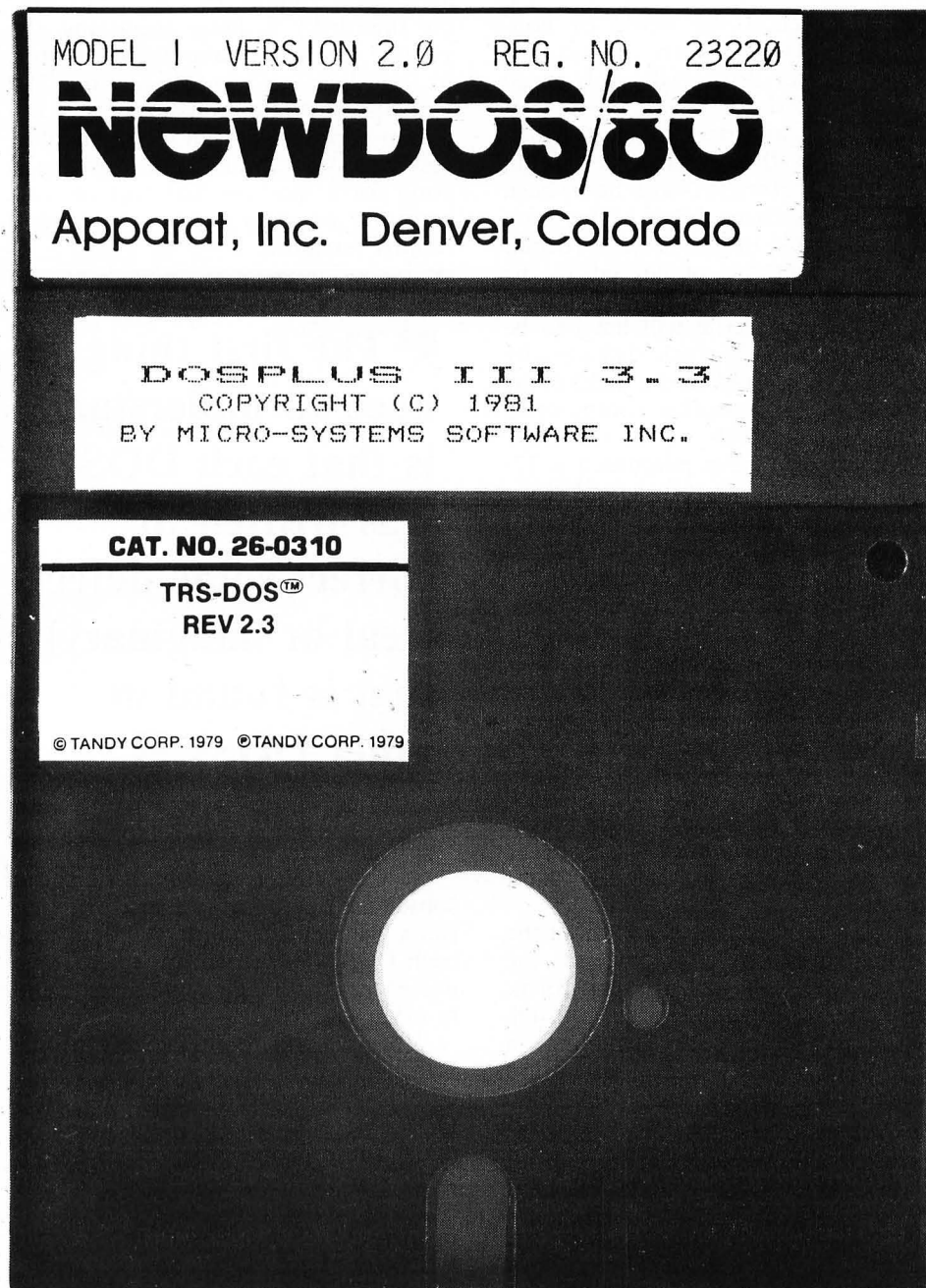
by Edward E. Umlor

This month we are going to look at a couple types of floppy drives and a DOS or two. I am somewhat opinionated in aspects of DOSs, but each DOS has some strong and some weak points. All drives discussed will be 5¼-inch drives unless I tell you otherwise.

The original drive that I am sure everyone remembers is the 35-track. This drive has a physical stop that allows the head carriage to travel a shortened distance toward the center of the disk. Track 00 is the track closest to the outside edge of the diskette and the highest numbered track is closest to the center of the diskette. We know that the smaller a circle's diameter, the shorter is its circumference. This means that each successive track has a shorter distance to travel. This also means that the amount of magnetic material encountered for a given bit of information is also less with the higher numbered tracks. The less material, the less energy is stored and the harder it is to detect the information. The head was stopped at track 35 because you could not obtain reliable data reads beyond that point on the diskette.

The next advance was to change the head so that reliable data reads could be obtained at these lower signal levels. This was done and the 40-track drive was announced. The physical stop was moved closer to the center of the diskette allowing the head to step an additional five track spaces. The distance from track to track is the same as the 35-track drive. This means that a 40-track diskette can be read (except for the five inner-most tracks) by a 35-track drive, and a 35-track diskette can be read by a 40-track drive. The capability of the head and the placement of the physical stop are the only differences between the 35- and 40-track drives.

The next step was to add an additional index hole and write-protect detector to the drive. This means that you can insert the diskette with a write-protect notch up and it will allow you to write/read the diskette. You can then remove the diskette, reinsert it with the write-protect notch down, and still be able to write/read the diskette. This type of drive is called a floppy



drive. You can flip the diskette over and use the other side. The biggest advantage of this type of drive is buying half as many diskettes for the same amount of data storage. The floppy drive can be 40-track or 80-track. The most popular floppy drive has been the 40-track.

The next head improvement was to allow it to handle the higher frequen-

cies required by the double-density format. (As we already know, single-density is ten sectors per track and double-density is 18 sectors per track on the TRS-80® Model I.) These advancements in head design have allowed the 77- and 80-track drives to come into existence. The diskette media manufacturers have been improving their formulations as well and

the combination of these two improvements has vastly increased the reliability of data transfer between disk and computer.

The 77-track drive was the first to come out with a 1/2 step head positioning motor (as compared to 35/40-track drives). This drive covers just a little more distance toward the center of the diskette than a 35-track drive. It is also capable of single- or double-density operation.

Here we enter the world of non-compatibility between drives. A 77-track disk drive will only read 77-track diskettes. It will not acknowledge any data on a 35-track, 40-track, or 80-track diskette. (I am speaking of diskettes that have been formatted by these drives.) There are DOSs available in the 77-track format and one must be purchased if the 77-track drive is to be your only drive. All program material will have to be entered from a 77-track diskette, by keyboard and saved to the disk, or by an interface with another computer to be saved on the disk.

Many people have purchased a 77- or 80-track drive as their initial drive without realizing that a special operating diskette had to be ordered with it. This has caused feelings of being cheated on the part of some consumers, but the truth of the matter is they received exactly what they ordered. If you've decided to purchase a 77- or 80-track drive as your initial drive, please LET THE PERSON TAKING THE ORDER KNOW so that the order can be marked for a special DOS diskette. These DOS diskettes are usually available from the same supplier you are talking to (at a price).

All that has been said here about the 77-track drive is also true of the 80-track drive in its original form. There are some suppliers of 80-track disk drives that are installing a 40/80-track switch. This modification allows you to read and copy program/data files from 40-track diskettes. You might consider this an ideal situation for an initial drive for your system.

Now I guess it is time to talk a little about the hard disk drive. The system I am familiar with is the Corvus 11T for the TRS-80® Model I. This system is using a modified NEWDOS+ and formats the hard disk into the equivalent of 94 drives of 35 or 40 tracks (depending on which DOS you ordered). The disk is formatted with the same gap and data format as the soft-sectored diskette with the addition of a volume marker. Each volume is the equivalent of a full drive. Corvus is now offering

NEWDOS80 Ver. 1.0 and NEWDOS80 Ver. 2.0 (by the time this article appears, barring problems). These new DOSs will allow the user to specify volume size up to a maximum value (at least one megabyte). This will be a much more efficient use of the hard disk. Corvus claims that a search and load of a file (volume unspecified) takes a maximum of ten seconds. I have found that asking for a program located on volume 93 (the last volume on the disk) without specifying the volume takes only about five seconds to find, load, and start execution. The high rate of data transfer is very apparent in programs that require a lot of disk accesses. The hard disk system is good for a business that requires vast amounts of data to be stored and rapidly retrieved.

■ The first thing we need to understand is that each DOS was written to correct some defect (real or imaginary) that is found in other DOSs. ■

Now it's time to share a little about some disk operating systems. The first thing we need to understand is that each DOS was written to correct some defect (real or imaginary) that is found in other DOSs.

When we talk about TRS-80® Model I DOSs, we have to mention Randy Cook, who wrote the original Tandy DOS. Here is where I am going to make some people mad, but Randy's DOS was the first one for the Tandy microcomputers and therefore is the inspiration behind all those that followed. Each DOS has its strong points and its weak points. You have to select a DOS for the Model I to meet your specific needs and that will require some research on your part. I think I have found the theme for next month: a method of system selection for the new computer buyer.

The first DOS to look at is the Ver. 2.3 of Tandy for the Model I. I will admit to not having used this DOS sufficiently to know all the points. This

DOS is considered a basic DOS (minimal) by myself. It functions OK but leaves a lot to be desired. It is all right as a user's DOS, but as a repairman and programmer (?) it is cumbersome for me to use. It does not allow for mixing of 35-, 40-, 77-, or 80-track 5¼-inch drives or the addition of 8-inch drives with 5¼. It is a 35-track operating system and any deviation from this requires modification of the DOS. This can be done through a means called PATCHING. Percom sells a PATCH PAK for changing the 2.3 DOS to 40-track or 77-track.

When it comes to VTOS/LDOS, I will have to claim total ignorance. I have not used either one. The words I have heard are: It is a good DOS for mixing drives, chaining (*setting up a sequential file for the execution of different programs*), and is strong in the BASIC programming editing functions. There are several magazines that have reviewed these DOSs and information on them can be found. (Reading magazines falls into spare time for me and I haven't found any of that in a long time.)

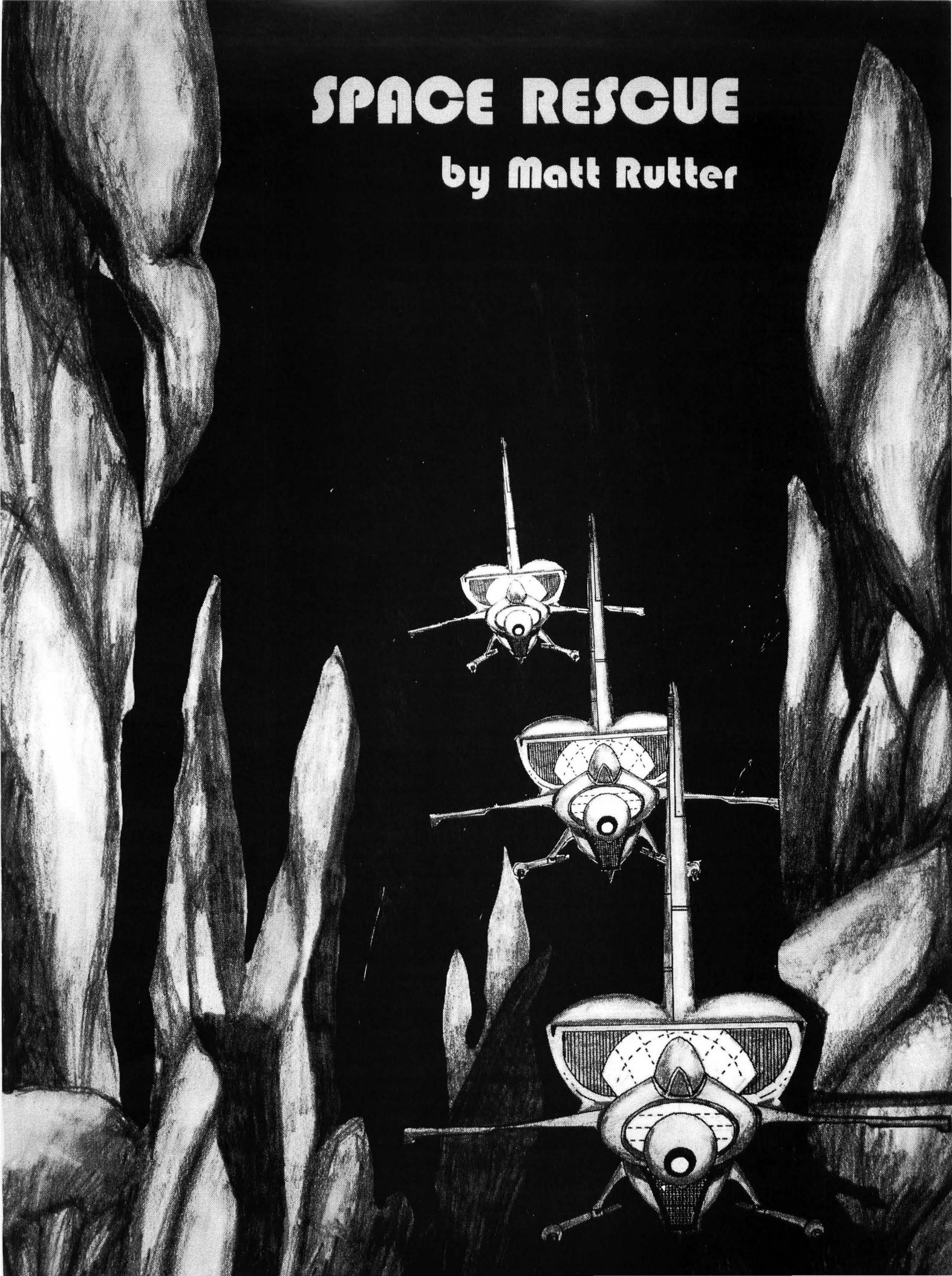
The DOS I am most familiar with is NEWDOS80 Ver. 1.0 (and have just started working with Ver. 2.0). This is a programmer's DOS in my humble opinion. It contains many features that help the programmer to rapidly build and edit a BASIC program. Version 1.0 allows for the intermixing of drives and if zapped to double-density by *Zap 2.0*, will allow the intermixing of single- and double-density formats on a multi-drive system. Version 2.0 will allow the intermixing of drives, single-density, Model I double-density, and Model III double-density formats on a Model I with a doubler installed. I have found that the NEWDOS80s fill my DOS needs the best.

The intent here is not to review the DOSs, but to let you know some of the different ones available. DOSPLUS has been written to recognize only diskettes formatted by DOSPLUS and is not compatible with anyone else's format. I have used it some on the Model III TRS-80® and it does seem to be a fairly flexible DOS. It must be EMPHASIZED that you should write down your requirements for a DOS and then purchase the one that most closely fills the bill.

This is OLE GRANITE KNOGGIN saying so long and keep the letters (both pro and con) coming. Oh, by the way, DISKIES is a coined (by this dummy) name for all computer people (and nuts — I don't want to discriminate) who have or are in the process of buying disk drives for their computers. ☺

SPACE RESCUE

by Matt Rutter



***Space Rescue* is an arcade-style game for a 16K ATARI®, Apple, or TRS-80®. The Apple version requires Applesoft and game paddles, and the ATARI® version requires one joystick.**

The year is 2086. Just a few months ago, the United States launched an exploration party to search the planet Arcturus III for any signs of life. Our radar, however, has just picked up a huge meteor storm headed straight for that solar system which threatens the lives of all the people there. You are their only hope of survival. You must launch your two-person rocket from the mother ship orbiting around the planet, land at the one landing pad, rescue the people stranded there one at a time, and then return to the mother ship — all while trying to avoid crashing your fragile rocket into one of the deadly meteors which can easily destroy it.

When the game starts, the computer will show the mother ship moving back and forth at the top of the screen with a cluster of meteors right below it. When you think that the ship is right over a path through the meteors, press the fire button on the joystick (ATARI®), the paddle button (Apple), or the spacebar (TRS-80®) to launch your rocket. Then you must guide the rocket down to the landing pad by moving your joystick, paddle, or left/right arrow keys, being careful not to collide with any of the meteors. (Note: The TRS-80® version has two speeds of horizontal movement. The ship normally moves TWO spaces at a time. For more delicate maneuvers such as landing on the pad or docking, the CLEAR key acts as a "fine tuner," moving the ship only one space at a time.) To slow yourself down, press the fire button, paddle button, or spacebar to apply thrust. After you have landed, one of the people will run over to your ship, and then

you must make your ascent to the mother ship and dock with her, again carefully avoiding the deadly meteors. On the way up, pressing the fire button, paddle button, or spacebar will launch a missile which can destroy a meteor, for which you can receive points.

If one of your rockets collides with a meteor, then it is destroyed. The game continues until all three of your rockets are destroyed, at which point the game is over. When all of the people are gone from the bottom of the screen, the computer will award you 50 bonus points for each person safely brought to the mother ship, and will then give you six more people to save. If you succeed in rescuing all six, you will be awarded one bonus ship. During the game, the score is displayed at the bottom of the screen underneath the landing pad.

Variables

- A, B: Used in determining whether or not a collision has occurred.
- D: Direction in which mother ship is travelling (1 or -1).
- D1: Difficulty level.
- G: Missile-launched flag.
- L: Position of landing pad at bottom of screen.
- P: Value returned from joystick, paddle, or keyboard input commands.
- P1: Number of people left at bottom of screen.
- P2: Number of people safely brought to mother ship.
- S: Position of mother ship at top of screen.
- SC: Score.
- SL: Number of ships left.
- U: Rocket-going-up flag.
- X, Y: Position of rocket.
- XM, YM: Position of missile.

```

$$$$$$$$$$$$$$$$$$$$
$ Applesoft BASIC $
$ 'SPACE RESCUE' $
$ AUTHOR: Matt Rutter $
$ TRANSL: Alan J. Zett $
$ (c) 1982 SoftSide $
$$$$$$$$$$$$$$$$$$$$

```

```

10 T1 = 895:T2 = 128:T3 = 984:T4 =
   8:T5 = 16: GOSUB 1100: GOTO
   15

```

```

13 V = INT (V):H = INT (H):AZ =
   T1 + V * T2 + H - T3 * ((V >
   T4) + (V > T5)): RETURN

```

Initialize variables.

```

15 D1 = 0:SC = 0:SL = 3:D% = CHR%
   (10):B% = CHR% (8)

```

```

20 TEXT : HOME :S = 35:D = - 1:
   P1 = 6:P2 = 0

```

```

25 FOR I = 22 TO 24: HTAB 1: VTAB
   I: PRINT "X";: HTAB 39: PRINT
   "X";: NEXT I

```

Draw the landing pad and put the meteors on the screen.

```

30 L = INT ( RND (1) * 23 + 6): HTAB
   L: VTAB 22: INVERSE : PRINT
   SPC( 3): HTAB L - 1: VTAB 2
   3: PRINT SPC( 5): HTAB L -
   2: VTAB 24: PRINT SPC( 7);:
   B = 0:G = 0: NORMAL

```

```

40 FOR I = 1 TO D1 * 10 + 30: HTAB
   RND (1) * 35 + 2: VTAB RND
   (1) * 15 + 4: PRINT "*":: NEXT
   I: HTAB L: VTAB 24: PRINT SC
   ;: FOR AZ = 1 TO SL: PRINT CHR%
   (7):: NEXT

```

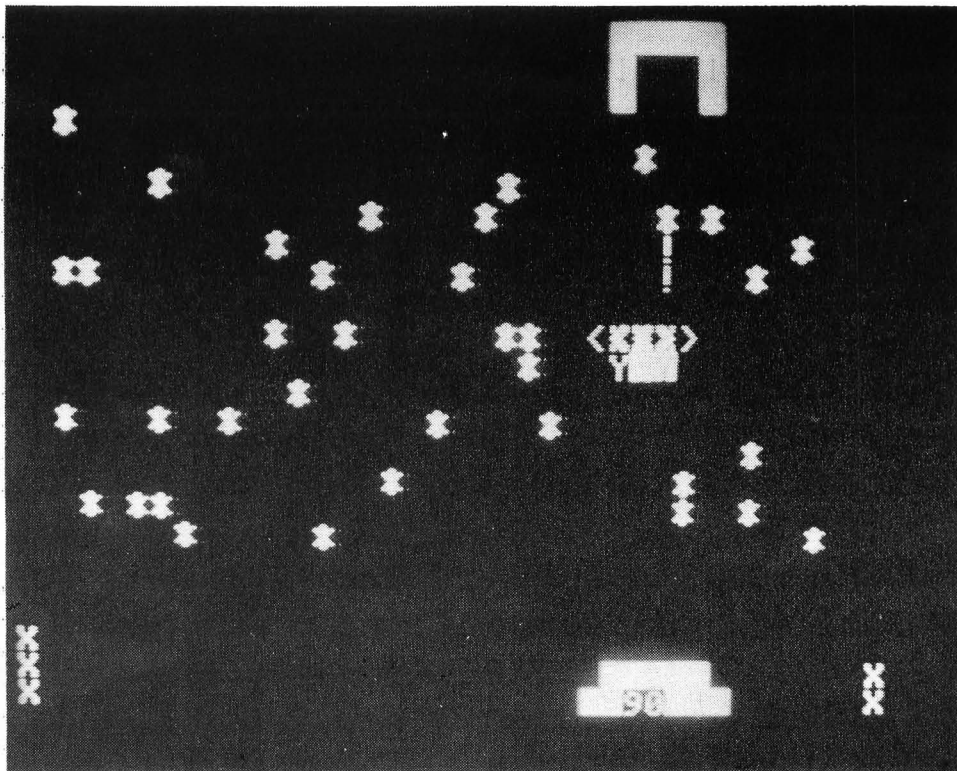
Move mother ship back and forth.

```

50 HTAB S: VTAB 1: PRINT SPC( 5
   )D%B%B%B%B% SPC( 5)D%B%B%B
   %B%B% SPC( 5);:S = S + D
60 INVERSE : HTAB S: VTAB 1: PRINT
   CHR% (92)"---"/D%B%B%B%B%
   "I";: NORMAL : PRINT "<X>";:
   INVERSE : PRINT "I"D%B%B%B%
   B%B%"/---" CHR% (92);: IF S <
   4 OR S > 34 THEN D = - D
65 NORMAL : IF PEEK ( - 16287) >
   127 THEN 80
70 FOR W = 1 TO 100: NEXT W: GOTO
   50
80 HTAB S + 2: VTAB 3: PRINT " "
   : FOR W = 1 TO 210: NEXT W: HTAB
   S + 1: VTAB 3: PRINT SPC( 3
   ):X = S + 2:Y = 2

```

Check for player input.



```

100 U = 0:X1 = X:P = PDL (0): IF
   (P < 50 OR P > 200) AND Y >
   2.5 THEN GOSUB 300
120 Y1 = Y + 1: IF PEEK ( - 1628
   7) > 127 THEN Y1 = Y + .5

```

Check for collision.

```

130 FOR I = X1 - 1 TO X1 + 1:H =
   I:V = Y1 + .5: GOSUB 13:A =
   PEEK (AZ): IF A = 32 OR A =
   170 THEN 500
133 IF B = 32 OR B = 170 THEN 50
   0
135 NEXT I

```

Move ship and display flame if appropriate.

```

140 HTAB X - 1: VTAB Y + .5: PRINT
   SPC( 3): HTAB X: VTAB Y + 1
   .5: PRINT " ":X = X1:Y = Y1:
   HTAB X - 1: VTAB Y + .5: PRINT
   "<X>"
145 H = X:V = Y + 1.5: GOSUB 13:B
   = PEEK (AZ): IF PEEK ( -
   16287) > 127 THEN HTAB X: VTAB
   Y + 1.5: FLASH : PRINT "Y": FOR
   AZ = 1 TO 7: & TY * 8,2: & T
   Y * 3,2: NEXT : NORMAL

```

Check for successful landing.

```

150 IF X = L + 1 AND INT (Y + .

```

```

5) = 21 THEN HTAB X: VTAB Y
   + 1.5: INVERSE : PRINT " ":
   NORMAL :SC = SC + 50: HTAB
   L: VTAB 24: PRINT SC:: GOTO
   400
155 IF INT (Y) > 21 THEN 500
160 GOTO 100

```

Erase old meteors and display new ones.

```

200 X = L + 1:Y = 21: FOR I = 4 TO
   19: HTAB 2: VTAB I: PRINT SPC(
   37): NEXT I
205 FOR I = 1 TO D1 * 15 + 40: HTAB
   RND (1) * 35 + 2: VTAB RND
   (1) * 15 + 4: PRINT "*":: NEXT
   I:C = 0
207 HTAB L: VTAB 22: PRINT SPC(
   3)

```

Make launching sound, check for input, and move ship up one space.

```

210 FOR AZ = 1 TO 7: & TY * 8,2:
   & TY * 3,2: NEXT :X1 = X:Y1
   = Y:P = PDL (0): IF P < 50
   OR P > 200 THEN GOSUB 300
220 C = C + 1: IF C = 3 THEN C =
   0:Y1 = Y - 1

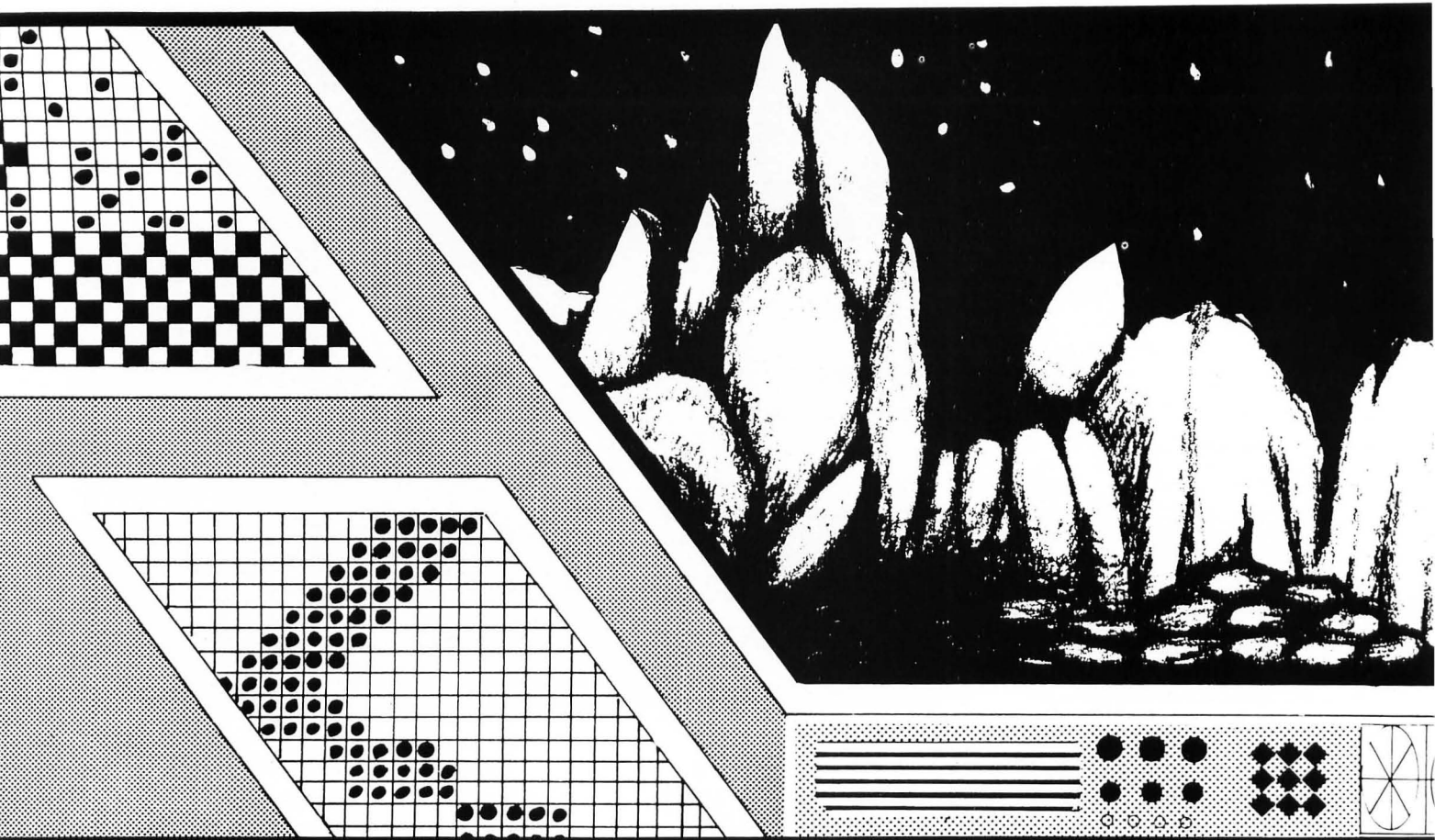
```

Check for collision.

```

225 FOR I = X1 - 1 TO X1 + 1:H =

```



```

I:V = Y1 + .5: GOSUB 13:A =
PEEK (AZ): IF A = 170 THEN
500
227 NEXT I

Move rocket and check for input.

230 HTAB X - 1: VTAB Y + .5: PRINT
SPC( 3): HTAB X: VTAB Y + 1
.5: PRINT " ":X = X1:Y = Y1:
HTAB X - 1: VTAB Y + .5: PRINT
"<X>": HTAB X: VTAB Y + 1.5:
FLASH : PRINT "Y"; : NORMAL

240 IF PEEK ( - 16287) > 127 AND
G = 0 AND Y > 5.5 THEN GOSUB
340
250 IF G = 1 THEN GOSUB 350
255 IF INT (Y) = 2 THEN 270
260 GOTO 210

Check for successful docking with
mother ship.

270 IF X < > S + 2 THEN 500
275 P2 = P2 + 1
280 HTAB 5: VTAB 3: INVERSE : PRINT
"/---" CHR$ (92); : NORMAL : FOR

```

```

I = 4 TO 24: HTAB 2: VTAB I:
PRINT SPC( 37): NEXT I

If all people are gone from the
bottom of the screen, then increase
difficulty level and branch to bonus
routine.

285 IF P1 = 0 THEN P1 = 6:D1 = D
1 + 1: GOTO 650
290 GOTO 30

Subroutine to move ship according
to input.

300 P = PDL (0): IF P < 50 AND X
> 3 THEN X1 = X - 1
310 IF P > 200 AND X < 37 THEN X
1 = X + 1
320 RETURN

Launch missile.

340 FOR AZ = 10 TO 100 STEP 10: &
TAZ,3: NEXT :G = 1:XM = X:YM =
Y

Move missile and check for collision
with meteors.

```

```

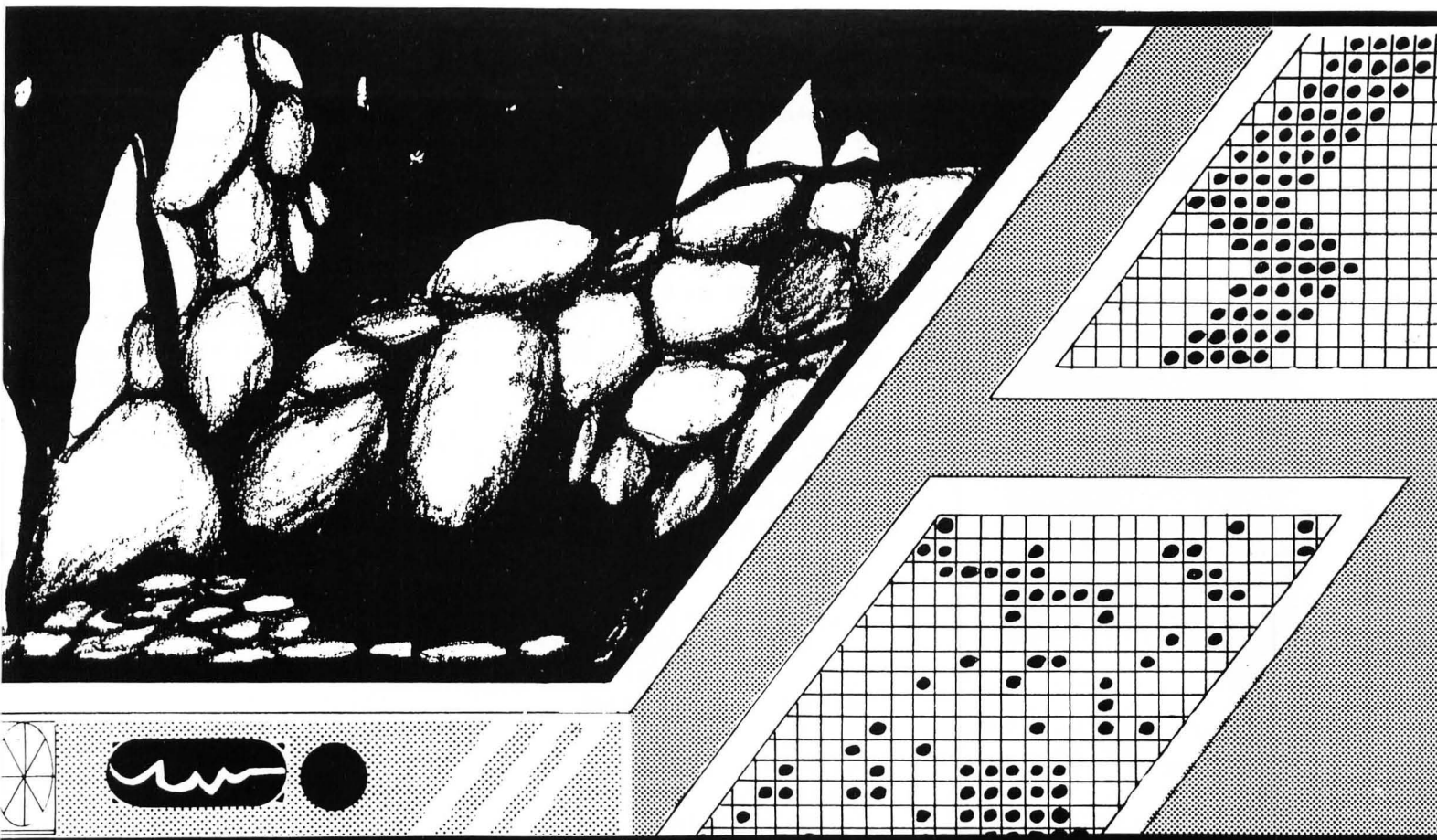
350 HTAB XM: VTAB YM + .5: PRINT
" ":YM = YM - 1
360 H = XM:V = YM + .5: GOSUB 13:
A = PEEK (AZ): IF A = 170 THEN
HTAB XM: VTAB YM + .5: PRINT
" ";G = 0:SC = SC + 20: HTAB
L: VTAB 24: PRINT SC; : RETURN

370 HTAB XM: VTAB YM + .5: PRINT
"!": IF YM < 5 THEN HTAB XM
: VTAB YM + .5: PRINT " ":G =
0
380 RETURN

Successful landing routine. Wave
arms of the next person and move
him over to rocket.

400 Y = 28 - P1:X = 39:X1 = 35:X2
= L + 5: IF P1 < 4 THEN Y =
25 - P1:X = 1:X1 = 2:X2 = L -
3
410 FOR I = 1 TO 4: FOR J = 43 TO
88 STEP 45: HTAB X: VTAB Y: PRINT
CHR$ (J); : FOR W = 1 TO 150
: NEXT W: NEXT J
420 HTAB X1: VTAB Y: PRINT "YAY!
"; : IF I / 2 = INT (I / 2) THEN

```



```

HTAB X1: VTAB Y: PRINT SPC(
4) : FOR AZ = 1 TO 2: FOR AY =
77 TO 7 STEP - 6: & TAY,4: NEXT
: FOR AY = 7 TO 77 STEP 6: &
TAY,4: NEXT : NEXT
430 NEXT I: HTAB X: VTAB Y: PRINT
" ";
440 FOR I = X + SGN (X1 - X) TO
X2 STEP SGN (X2 - X)
450 FOR J = 43 TO 88 STEP 45: HTAB
I: VTAB 24: PRINT CHR$( J);
: FOR W = 1 TO 100: NEXT W
455 & T100,2: & T200,2: NEXT J: HTAB
I: VTAB 24: PRINT " "; NEXT
I
460 X = I + SGN (X1 - X2):Y = 24
: FOR I = 1 TO 2:X = X + SGN
(L - X1):Y = Y - 1
465 FOR J = 43 TO 88 STEP 45: HTAB
X: VTAB Y: PRINT CHR$( J);;
FOR W = 1 TO 100: NEXT W: HTAB
X: VTAB Y: PRINT " "; NEXT
I
470 P1 = P1 - 1:U = 1: GOTO 200

```

Explosion routine.

500 REM

```

520 X1 = X - 2:X2 = X + 1:X0 = X;
XS = 2: HTAB X - 1: VTAB Y +
.5: PRINT SPC( 3): HTAB X: VTAB
Y + 1.5: PRINT " "
530 FOR I = Y TO 23: HTAB X1: VTAB
I: PRINT "<<("; HTAB X2: VTAB
I: PRINT ")>>"; IF U = 1 THEN
HTAB X0: VTAB I: PRINT "X";
540 FOR AZ = 1 TO 17: & T RND ( 1
) & 255, INT ( RND ( 1) & 2) +
1: NEXT :XS = XS - .2: IF XS
< 0 THEN XS = 0
550 HTAB X1: VTAB I: PRINT SPC(
2): HTAB X2: PRINT SPC( 2):
IF U = 1 THEN HTAB X0: VTAB
I: PRINT " ";
560 X1 = X1 - XS: IF X1 < 2 THEN
X1 = 37
570 X2 = X2 + XS: IF X2 > 37 THEN
X2 = 2
580 NEXT I:SL = SL - 1: IF SL =
0 THEN 600
590 GOTO 280

```

End of game. Print score and game-over message and wait for input.

600 HOME : HTAB 14: VTAB 11: INVERSE

```

: PRINT "& GAME OVER &": NORMAL
: HTAB 12: VTAB 13: PRINT "Y
OUR SCORE IS ";SC
610 HTAB 7: VTAB 15: FLASH : PRINT
" PRESS RETURN TO PLAY AGAIN
";: NORMAL
620 POKE - 16368,0: CALL - 756
: GOTO 15

```

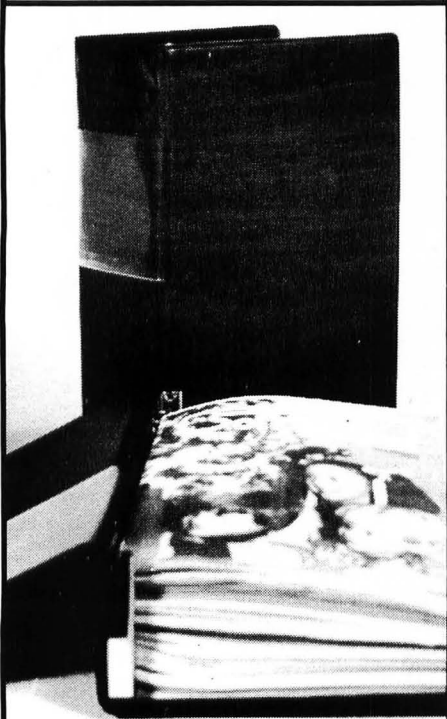
Bonus routine. Awards 50 points for each person safely brought to mother ship.

```

650 HOME
660 HTAB 16: VTAB 11: FLASH : PRINT
"& BONUS &": HTAB 16: VTAB 1
3: NORMAL : PRINT "SCORE=";S
C
670 FOR I = 1 TO P2: HTAB 13 + I
& 2: VTAB 15: PRINT "X":SC =
SC + 50: HTAB 22: VTAB 13: PRINT
SC
680 FOR AZ = 1 TO 10: & TI & 15 +
30,3: & TI & 15 + 60,3: NEXT
: FOR W = 1 TO 300: NEXT W: NEXT
I
690 FOR AZ = 1 TO 100: & T100 -
AZ,2: & TAZ,2: NEXT : IF P2 =

```

Protect Your Investment!



With SoftSide™ Vinyl Binders

Protect your *SoftSide* back issues (combined editions) with these sturdy binders. Covered with durable wood-grain vinyl, each 8½ x 11 inch binder has an inside pocket and clear sleeve on the spine which you can label for easy identification. Each binder holds 12 issues.

8½ x 11..... \$7.95

SoftSide
6 South Street Milford NH 03055

110

```
6 THEN HTAB 11: VTAB 17: INVERSE
: PRINT "### BONUS SHIP! ###"
": NORMAL :SL = SL + 1: FOR
P2 = 1 TO 4: FOR AZ = 80 TO
185 STEP 7: & TAZ,2: & TAZ +
50,2: NEXT : NEXT
```

Bonus ship routine. Awards a bonus ship if all six people are rescued.

```
695 FOR W = 11 TO 111 STEP 3: FOR
AZ = W TO W - 7 STEP - 1: &
TAZ,3: NEXT : NEXT : GOTO 20
```

Initialize sound routine.

```
1000 POK$ = "201,084,208,015,032,
177,000,032,248,230,138,072,
032,183,000,201,044,240,003,
076,201,222,032,177,000,032,
248,230,104,134,003,134,001,
133,000"
```

```
1010 FOR I = 1 TO 35: POKE I + 7
67, VAL ( MID$ (POK$,I * 4 -
3,I * 4 - 1)): NEXT I
```

```
1020 POK$ = "170,160,001,132,002,
173,048,192,136,208,004,198,
001,240,007,202,208,246,166,
000,208,239,165,003,133,001,
198,002,208,241,096"
```

```
1030 FOR I = 1 TO 33: POKE I + 8
02, VAL ( MID$ (POK$,I * 4 -
3,I * 4 - 1)): NEXT I
```

```
1040 POKE 1013,76: POKE 1014,0: POKE
1015,3: FOR AZ = 1 TO 5000: NEXT
: RETURN
```

```
1100 HGR : POKE - 16302,0: HOME
: FLASH
```

Draws opening display on the screen.

```
1110 FOR X = 1 TO 39 STEP 2: COLOR=
0: PLOT X,0: PLOT X,44: COLOR=
2: PLOT X,1: PLOT X,45: NEXT
```

```
1120 FOR Y = 0 TO 43 STEP 4: COLOR=
0: PLOT 1,Y: PLOT 39,Y: COLOR=
2: PLOT 1,Y + 1: PLOT 39,Y +
1: NEXT
```

```
1140 VTAB 5: HTAB 7: PRINT SPC(
5): HTAB 13: PRINT SPC( 5):
HTAB 19: PRINT SPC( 5): HTAB
25: PRINT SPC( 5): HTAB 31:
PRINT SPC( 5)
```

```
1142 VTAB 6: HTAB 7: PRINT " ";
HTAB 13: PRINT " "; HTAB 1
7: PRINT " "; HTAB 19: PRINT
" "; HTAB 23: PRINT " "; HTAB
25: PRINT " "; HTAB 31: PRINT
" "
```

```
1144 HTAB 7: PRINT SPC( 5): HTAB
13: PRINT " "; HTAB 17: PRINT
```

```
" "; HTAB 19: PRINT SPC( 5
): HTAB 25: PRINT " "; HTAB
31: PRINT SPC( 5)
1146 VTAB 8: HTAB 11: PRINT " ";
: HTAB 13: PRINT SPC( 5): HTAB
19: PRINT " "; HTAB 23: PRINT
" "; HTAB 25: PRINT " "; HTAB
31: PRINT " "
1148 HTAB 11: PRINT " "; HTAB 1
3: PRINT " "; HTAB 19: PRINT
" "; HTAB 23: PRINT " "; HTAB
25: PRINT " "; HTAB 31: PRINT
" "
1150 HTAB 7: PRINT SPC( 5): HTAB
13: PRINT " "; HTAB 19: PRINT
" "; HTAB 23: PRINT " "; HTAB
25: PRINT SPC( 5): HTAB 31:
PRINT SPC( 5)
1152 VTAB 14: HTAB 4: PRINT SPC(
5): HTAB 10: PRINT SPC( 5):
HTAB 16: PRINT SPC( 5): HTAB
22: PRINT SPC( 5): HTAB 28:
PRINT " "; HTAB 32: PRINT
" "; HTAB 34: PRINT SPC( 5
)
1154 VTAB 15: HTAB 4: PRINT " ";
: HTAB 8: PRINT " "; HTAB 1
0: PRINT " "; HTAB 16: PRINT
" "; HTAB 22: PRINT " "; HTAB
28: PRINT " "; HTAB 32: PRINT
" "; HTAB 34: PRINT " "
1156 HTAB 4: PRINT " "; HTAB 8:
PRINT " "; HTAB 10: PRINT
SPC( 5): HTAB 16: PRINT SPC(
5): HTAB 22: PRINT " "; HTAB
28: PRINT " "; HTAB 32: PRINT
" "; HTAB 34: PRINT SPC( 5
)
1158 VTAB 17: HTAB 4: PRINT SPC(
5): HTAB 10: PRINT " "; HTAB
20: PRINT " "; HTAB 22: PRINT
" "; HTAB 28: PRINT " "; HTAB
32: PRINT " "; HTAB 34: PRINT
" "
1160 HTAB 4: PRINT " "; HTAB 7:
PRINT " "; HTAB 10: PRINT
" "; HTAB 20: PRINT " "; HTAB
22: PRINT " "; HTAB 28: PRINT
" "; HTAB 32: PRINT " "; HTAB
34: PRINT " "
1162 HTAB 4: PRINT " "; HTAB 8:
PRINT " "; HTAB 10: PRINT
SPC( 5): HTAB 16: PRINT SPC(
5): HTAB 22: PRINT SPC( 5):
HTAB 28: PRINT SPC( 5): HTAB
34: PRINT SPC( 5)
1164 NORMAL : POKE - 16303,0: GOTO
1000
```



```

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Atari BASIC $
$ 'SPACE RESCUE' $
$ AUTHOR: Matt Rutter $
$ (c) 1982 SoftSide $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

```

In the ATARI® version lines 30, 50, 60, 80, 280, and 370 contain graphics and/or control characters.

```
10 GOSUB 1100:GOSUB 1000
```

Initialize variables. Set character base register (POKE location 756) to beginning of altered character set, and display six people at the bottom of the screen.

```

15 D1=0:SC=0:SL=3
20 GRAPHICS 0:POKE 756,R:POKE 752,1:SETCOLOR 2,0,0:S=34:D=-1:P1=6:P2=0
25 FOR I=21 TO 23:POSITION 0,I:?"f";:POSITION 38,I:?"f";:NEXT I

```

Draw the landing pad and put the meteors on the screen.

```

30 L=INT(23*RND(1)+5):POSITION L,21:?"_":POSITION L-1,22:?"_":POSITION L-2,23:?"_":B=0:G=0
40 FOR I=1 TO D1*10+30:POSITION RND(1)*35+1,RND(1)*15+3:?"CHR$(20)";:NEXT I:POSITION L,23:?"SC":POKE 77,0
45 FOR A=1 TO SL:SOUND 0,37,10,10:FOR W=1 TO 50:NEXT W:SOUND 0,0,0,0:NEXT A
46 REM In lines 50-80 & 280 upper-case and symbols in PRINT statements correspond to following:
47 REM '=' = 'ESC' 'CTRL' '='
48 REM '+' = 'ESC' 'CTRL' '+'
49 REM 'B' = 'CTRL' 'B'
50 REM 'F' = 'CTRL' 'F'
51 REM 'G' = 'CTRL' 'G'
52 REM 'M' = 'CTRL' 'M'
53 REM 'N' = 'CTRL' 'N'

```

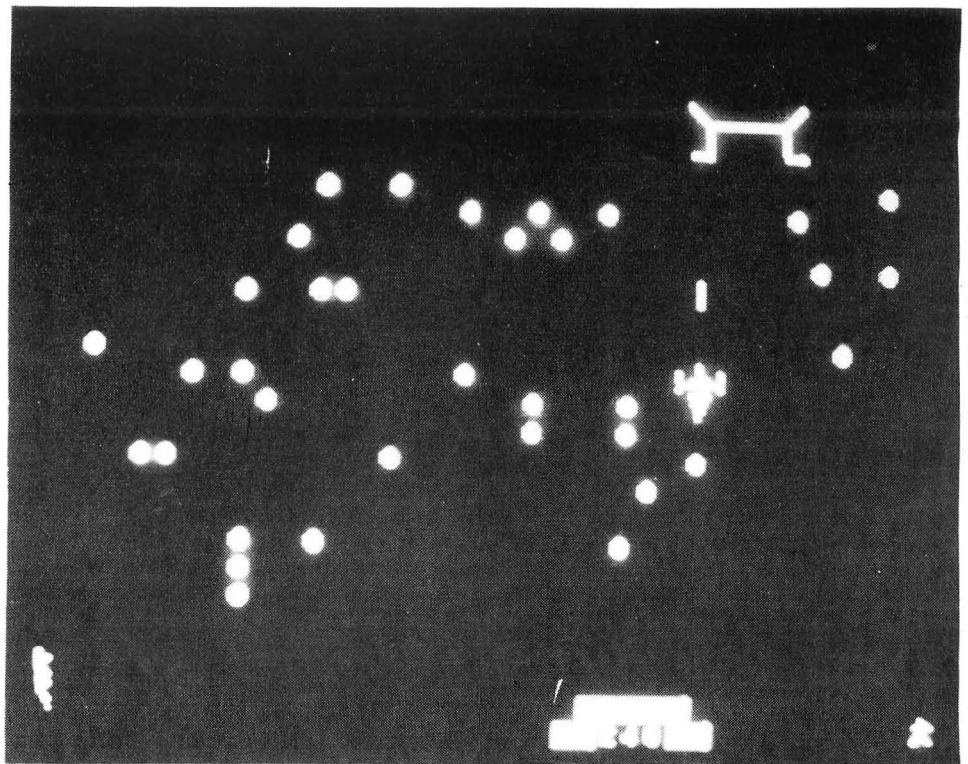
Move mother ship back and forth.

```

50 POSITION S,0:?"+++++ "+"
55 "":S=S+D
60 POSITION S,0:?"GNNF+++++BabcV+++MMH";:IF S<3 OR S>33 THEN D=-D
70 FOR W=1 TO 25:NEXT W:GOTO 50
80 POSITION S+2,2:?"M":FOR W=1 TO 50:NEXT W:POSITION S,2:?"M":X=S+2:Y=1

```

Check for player input.



```

100 U=0:X1=X:P=STICK(0):IF P<12 AND Y>1.5 THEN GOSUB 300
120 Y1=Y+1:P=STRIG(0):IF P=0 THEN Y1=Y+0.5

```

Check for collision.

```

130 FOR I=X1-1 TO X1+1:LOCATE I,Y1,A:IF A=20 OR A=160 THEN 500
133 IF B=20 OR B=160 THEN 500
135 NEXT I

```

Move ship and display flame if appropriate.

```

140 POSITION X-1,Y:?" ":POSITION X,Y+1:?" ":X=X1:Y=Y1:POSITION X-1,Y:?"abc"
145 LOCATE X,Y+1,B:IF P=0 THEN POSITION X,Y+1:?"d":SOUND 1,20,8,15
147 IF P=1 THEN SOUND 1,0,0,0

```

Check for successful landing.

```

150 IF X=L+1 AND INT(Y+0.5)=20 THEN POSITION X,Y+1:?"CHR$(160)":SC=SC+50:POSITION L,23:?"SC":GOTO 400
155 IF Y>20 THEN 500
160 GOTO 100

```

Erase old meteors and display new ones.

```
200 X=L+1:Y=20:COLOR 32:FOR I=3 TO 18:
```

```

PLOT 1,I:DRAWTO 37,I:NEXT I
205 FOR I=1 TO D1*15+40:POSITION RND(1)*35+1,RND(1)*15+3:?"CHR$(20)";:NEXT I:
C=0
207 POSITION L,21:?" " "

```

Make launching sound, check for input, and move ship up one space.

```

210 SOUND 0,5*Y+90,8,8:X1=X:Y1=Y:P=STICK(0):IF P<12 THEN GOSUB 300
220 C=C+1:IF C=3 THEN C=0:Y1=Y-1

```

Check for collision.

```

225 FOR I=X1-1 TO X1+1:LOCATE I,Y1,A:IF A=20 THEN 500
227 NEXT I

```

Move rocket and check for input.

```

230 POSITION X-1,Y:?" ":POSITION X,Y+1:?" ":X=X1:Y=Y1:POSITION X-1,Y:?"abc":POSITION X,Y+1:?"d";
240 IF STRIG(0)=0 AND G=0 AND Y>4 THEN GOSUB 340
250 IF G=1 THEN GOSUB 350
255 IF Y=1 THEN 270
260 GOTO 210

```

Check for successful docking with mother ship.

```

270 IF X<>S+2 THEN 500
275 P2=P2+1

```

```
280 SOUND 0,0,0,0:POSITION S,2:? " MMM
  ":COLOR 32:FOR I=3 TO 23:PLOT 1,I:DRA
WTO 37,I:NEXT I
```

If all people are gone from the bottom of the screen, then increase difficulty level and branch to bonus routine.

```
285 IF P1=0 THEN P1=6:D1=D1+1:GOTO 650
290 GOTO 30
```

Subroutine to move ship according to input.

```
300 IF P>8 AND P<12 AND X>2 THEN X1=X-1
310 IF P<8 AND X<36 THEN X1=X+1
320 RETURN
```

Launch missile.

```
340 G=1:XM=X:YM=Y:FOR I=10 TO 100 STEP
  10:SOUND 1,1,10,10:NEXT I:SOUND 1,0,0,0
```

Move missile and check for collision with meteors.

```
350 POSITION XM,YM:? " ":YM=YM-1
360 LOCATE XM,YM,A:IF A=20 THEN POSITI
ON XM,YM:? " ":G=0:SC=SC+20:POSITION
L,23:? SC:;RETURN
370 POSITION XM,YM:? CHR*(124):IF YM<4
  THEN POSITION XM,YM:? " ":G=0
380 RETURN
```

Successful landing routine. Wave arms of the next person and move him over to rocket.

```
400 SOUND 1,0,0,0:Y=27-P1:X=38:X1=34:X
  2=L+5:IF P1<4 THEN Y=24-P1:X=0:X1=1:X2
  =L-3
410 FOR I=1 TO 8:FOR J=101 TO 103:POSI
  TION X,Y:? CHR*(J):;FOR W=1 TO 30:NEXT
```

```
W:NEXT J
420 POSITION X1,Y:? "YAY!";:IF I/2=INT
  (I/2) THEN POSITION X1,Y:? " ";
430 NEXT I:POSITION X,Y:? " ";
440 FOR I=X+SGN(X1-X) TO X2 STEP SGN(X
  2-X)
450 FOR J=101 TO 103 STEP 2:POSITION I
  ,23:? CHR*(J):;FOR W=1 TO 20:NEXT W
455 SOUND 0,90,8,15:SOUND 0,0,0,0:NEXT
  J:POSITION I,23:? " ":NEXT I
460 X=X+SGN(X1-X2):Y=23:FOR I=1 TO 2:X
  =X+SGN(L-X1):Y=Y-1
465 FOR J=101 TO 103 STEP 2:POSITION X
  ,Y:? CHR*(J):;FOR W=1 TO 20:NEXT W:NEX
  T J:POSITION X,Y:? " ":NEXT I
470 P1=P1-1:U=1:GOTO 200
```

Explosion routine.

```
500 SOUND 0,0,0,0:SOUND 1,75,8,15
520 X1=X-2:X2=X+1:X0=X:X5=2:V0=15:V1=1
  5:V2=15:POSITION X-1,Y:? " ":POSITIO
  N X,Y+1:? " "
530 FOR I=Y TO 22:POSITION X1,I:? "ah"
  ;:POSITION X2,I:? "ic";:IF U=1 THEN PO
  SITION X0,I:? "e";
540 SOUND 0,20,8,8:V0=SOUND 1,40,8,V1:SD
  UND 2,70,8,V2:V0=V0*0.7:V1=V1*0.75:V2=
  V2*0.78:X5=X5-0.2:IF X5<0 THEN X5=0
550 FOR W=1 TO 30:NEXT W:POSITION X1,I
  :? " ":;:POSITION X2,I:? " ":;:IF U=1
  THEN POSITION X0,I:? " "
560 X1=X1-X5:IF X1<1 THEN X1=36
570 X2=X2+X5:IF X2>36 THEN X2=1
580 NEXT I:FOR I=0 TO 3:SOUND 1,0,0,0:
  NEXT I:SL=SL-1:IF SL=0 THEN 600
590 GOTO 280
```

End of game. Print score and game-over message and wait for input.

```
600 GRAPHICS 0:POKE 752,1:POSITION 16,
  10:? "GAME OVER":POSITION 12,12:? "YOU
  R SCORE IS ";SC
610 POSITION 8,14:? "PUSH START TO PLA
  Y AGAIN"
```

```
620 IF PEEK(53279)=6 THEN 15
630 GOTO 620
```

Bonus routine. Awards 50 points for each person safely brought to mother ship.

```
650 FOR I=3 TO 23:PLOT 1,I:DRAWTO 37,I
  :NEXT I
660 POSITION 16,10:? "BONUS":POSITIO
  N 15,12:? "SCORE=";SC
670 FOR I=1 TO P2:POSITION 12+I*2,14:?
  "f":SC=SC+50:POSITION 21,12:? SC
680 SOUND 0,I*20+40,10,10:FOR W=1 TO 5
  0:NEXT W:NEXT I
690 FOR I=P2 TO 1 STEP -1:SOUND 0,I*20
  +40,10,10:FOR W=1 TO 50:NEXT W:NEXT I:
  SOUND 0,0,0,0
```

Bonus ship routine. Awards a bonus ship if all six people are rescued.

```
693 IF P2=6 THEN POSITION 10,16:? "###
  BONUS SHIP! ###";:SL=SL+1:FOR P2=1 TO
  4:FOR AZ=80 TO 185 STEP 6
696 SOUND 0,AZ,10,10:SOUND 0,AZ+50,10,
  10:NEXT AZ:NEXT P2:FOR W=11 TO 111 STE
  P 5:FOR AZ=W TO W-7 STEP -1
699 SOUND 0,AZ,10,10:NEXT AZ:NEXT W:SD
  UND 0,0,0,0:GOTO 20
```

Routine to alter the character set. First it loads the old character set from ROM starting at location 57344, writing it to RAM starting 2048 bytes or 8 pages from the top of memory. Then it POKES the new characters from the data statements into the RAM of the new character set. This alters the lower-case characters a-i.

```
1000 R=PEEK(106)-8:RM=R*256
1010 C=0:FOR I=0 TO 1023 STEP 30:C=C+1
  :IF C=3 THEN C=0
1020 SETCOLOR C,1,8:FOR J=I TO I+30:PO
  KE RM+J,PEEK(57344+J):NEXT J:SETCOLOR
  C,0,0:NEXT I:SETCOLOR 0,1,8
1030 FOR I=0 TO 8:FOR J=0 TO 7:READ A:
  POKE RM+(97+I)*8+J,A:NEXT J:NEXT I:RET
  URN
1040 REM DATA FOR NEW CHARACTERS, IN
  GROUPS OF 8
1050 DATA 0,0,12,12,12,15,15,12,24,24,
  24,60,255,255,255,0,0,0,48,48,48,240,2
  40,48
1060 DATA 255,255,126,126,60,60,24,24,
  219,126,60,24,24,60,102,195,24,24,255,
  24,24,60,102,195
1070 DATA 24,24,60,126,219,60,102,195,
```

Continental Adventures

4975 Brookdale Dept. 04
Bloomfield Hills, Mich. 48013
(313) 645-2140

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

The Ghost Tower — Combat with diabolical demons, 16K \$16.95

Town of Derango — Avenging the death of a father, 8K \$16.95

Talisman of Power — A search for the four keys of Gremlock, 16K... \$18.95

Super Shape Builder — A graphics game for creating your own pictures. Joysticks reqd. 8K \$14.95

```

48,48,48,112,240,240,0,12,12,12,14
,15,15,15,0
1100 GRAPHICS 3+16:FOR I=0 TO 2:SETCOL
OR I,0,0:NEXT I
1105 C=0:FOR I=0 TO 39:C=C+1:IF C=4 TH
EN C=1

```

Draws opening display on the screen.

```

1110 COLOR C:PLOT I,0:PLOT 39-I,21:NEX
T I
1120 C=0:FOR I=0 TO 21:C=C+1:IF C=4 TH
EN C=1
1130 COLOR C:PLOT 39,I:PLOT 0,21-I:NEX
T I
1140 COLOR 1:PLOT 9,4:DRAWTO 5,4:DRAW
O 5,6:DRAWTO 9,6:DRAWTO 9,9:DRAWTO 5,9
:PLOT 11,9:DRAWTO 11,4:DRAWTO 15,4
1142 DRAWTO 15,7:DRAWTO 11,7:PLOT 17,9
:DRAWTO 17,4:DRAWTO 21,4:DRAWTO 21,9:P
LOT 17,6:DRAWTO 21,6:PLOT 27,4
1144 DRAWTO 23,4:DRAWTO 23,9:DRAWTO 27
,9:PLOT 33,4:DRAWTO 29,4:DRAWTO 29,9:D
RAWTO 33,9:PLOT 29,6:DRAWTO 33,6
1150 PLOT 2,17:DRAWTO 2,12:DRAWTO 6,12
:DRAWTO 6,15:DRAWTO 2,15:PLOT 5,16:PL
O 6,17:PLOT 12,12:DRAWTO 8,12
1152 DRAWTO 8,17:DRAWTO 12,17:PLOT 8,1
4:DRAWTO 12,14:PLOT 18,12:DRAWTO 14,12
:DRAWTO 14,14:DRAWTO 18,14
1154 DRAWTO 18,17:DRAWTO 14,17:PLOT 24
,12:DRAWTO 20,12:DRAWTO 20,17:DRAWTO 2
4,17:PLOT 26,12:DRAWTO 26,17
1156 DRAWTO 30,17:DRAWTO 30,12:PLOT 36
,12:DRAWTO 32,12:DRAWTO 32,17:DRAWTO 3
6,17:PLOT 32,14:DRAWTO 36,14
1160 RETURN

```



GALACTIC CHASE™

The aliens have swept undefeated across the galaxy. You are an enterprising star ship captain—the final defender of space.

As the aliens attack, you launch a deadly barrage of missiles. Flankers swoop down on your position. Maneuvering to avoid the counterattack, you disintegrate their ships with your magnetic repellers.

As your skill improves, the attackers increase their speed. And as a last resort, the aliens use their invisible ray to slow the speed of your missile launcher.

GALACTIC CHASE provides Atari owners with the most challenging one or two person game in the galaxy.



Atari 400/800 16k. Written in machine language. Requires joysticks.

Payment: Personal Checks—allow three weeks to clear.

American Express, Visa, & Master Charge—include all numbers on card. Please include phone number with all orders. 24.95 for cassette or 29.95 for disk plus 2.00 shipping. Michigan residents add 4%.

Check the dealer in your local galaxy. Dealer inquiries encouraged.

Galactic Chase © 1981 Stedek Software.



SPECTRUM
COMPUTERS

Dept S.
26618 Southfield
Lathrup Village, MI. 48076
(313) 559-5252

```

#####
$   S-80 BASIC   $
$   'SPACE RESCUE' $
$   AUTHOR: Matt Rutter $
$   TRANSL: Alan J. Zett $
$   (c) 1982 SoftSide $
#####

```

```
10 CLS:GOTO1100
```

Initialize variables.

```

15 RANDOM:DEFINT A-X,Z:D1=0:SC=0:SL=3:U$="#####."
20 CLS:S=51:D=-1:P1=6:P2=0:SHIP%=CHR$(158)+CHR$(191)+CHR$(173)
25 FORI=13TO15:PRINT@I#64+1,"X";:PRINT@I#64+62,"X";:NEXT

```

Draw the landing pad and put the meteors on the screen.

```

30 L=RND(39)+8:PRINT@894+L,CHR$(160)CHR$(184)CHR$(190)STRING$(3,
143)CHR$(189)CHR$(180)CHR$(144);:PRINT@958+L,STRING$(9,191);:B=0
:G=0
40 FORI=1TOD1#10+15:PRINT@ (1+RND(9))#64+RND(63),CHR$(140);:NEXT;
PRINT@959+L,USINGU%;SC/1E5;:PRINT@0;:FORAZ=1TOSL:SOUND37,100:SO
UNDO,100:NEXT

```

Move mother ship back and forth.

```

50 S=S+D
60 PRINT@S-1," "CHR$(191)STRING$(4,179)CHR$(191)STRING$(4,179)CH
R$(191)" "CHR$(26)STRING$(12,24)" "CHR$(131)CHR$(191)CHR$(149)SH
IP%CHR$(170)CHR$(191)CHR$(131)" ";:IFB<20RS>50THEND=-D
65 IF(PEEK(14400)AND128)=128THENB0
70 FORW=1TO25:NEXTW:GOTO50
80 X=S+5:Y=1

```

Check for player input.

```

100 U=0:X1=X:P=PEEK(14400)AND96:IF(P=32ORP=64)ANDY>1.560SUB300
120 Y1=Y+1:IF(PEEK(14400)AND128)=128THENY1=Y+.5

```

Check for collision.

```

130 FORI=X1-1TOX1+1:A=PEEK(15360+I+(INT(Y1+.5)#64)):IFA=140THEN5
00
133 IFB=140THEN500
135 NEXTI

```

Move ship and display flame if appropriate.

```

140 PRINT@X-1+(INT(Y+.5)#64)," ";:PRINT@X+(INT(Y+1.5)#64)," ";
: X=X1:Y=Y1:PRINT@X-1+(INT(Y+.5)#64),SHIP%;
145 B=PEEK(15360+X+(INT(Y+1.5)#64)):IF(PEEK(14400)AND128)=128THE
NPRINT@X+(INT(Y+1.5)#64),"V";:PRINT@0;:FORAZ=1TOSL:SOUNDY#8,2:SO
UNDY#3,2:NEXT

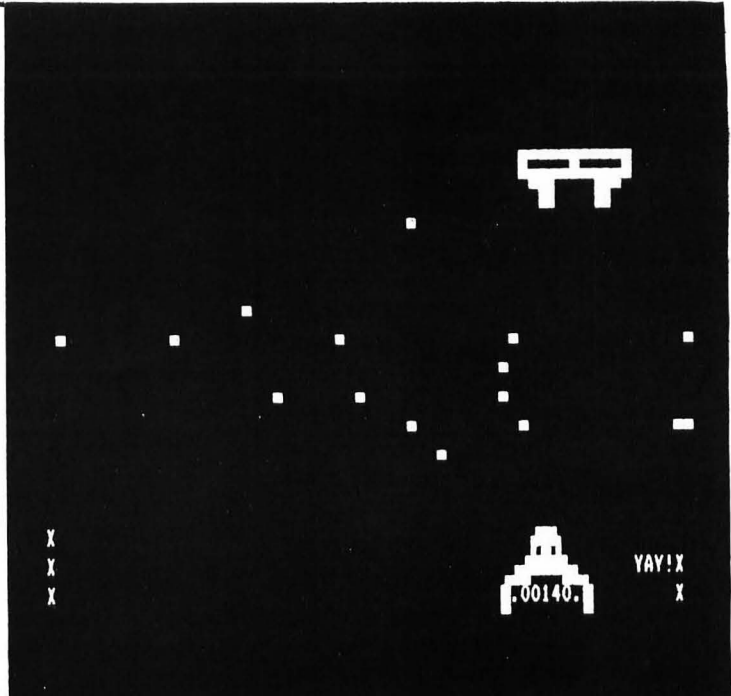
```

Check for successful landing.

```

150 IFX=L+2ANDINT(Y+.5)=13THENPRINT@X+(INT(Y+1.5)#64),CHR$(143);
:SC=SC+50:PRINT@L+959,USINGU%;SC/1E5;:GOTO400
155 IFINT(Y)>12THEN500
160 GOTO1100

```



Erase old meteors and display new ones.

```

200 X=L+2:Y=13:FORI=2TO12:PRINT@I#64,STRING$(64,32);:NEXT
205 FORI=1TOD1#15+20:PRINT@ (1+RND(9))#64+RND(63),CHR$(140);:NEXT
:PRINT@959+L,USINGU%;SC/1E5;
207 PRINT@894+L," ";

```

Make launching sound, check for input, and move ship up one space.

```

210 PRINT@0;:FORAZ=1TOSL:SOUNDY#8,2:SOUNDY#3,2:NEXT:X1=X:Y1=Y:P=
PEEK(14400)AND96:IFP=32ORP=64THENGOSUB300
220 C=C+1:IFC=3THENC=0:Y1=Y-1

```

Check for collision.

```

225 FORI=X1-1TOX1+1:A=PEEK(15360+I+(INT(Y1+.5)#64)):IFA=140THEN5
00
227 NEXT

```

Move rocket and check for input.

```

230 PRINT@X-1+(INT(Y+.5)#64)," ";:PRINT@X+(INT(Y+1.5)#64)," ";
: X=X1:Y=Y1:PRINT@X-1+(INT(Y+.5)#64),SHIP%;:PRINT@X+(INT(Y+1.5)#6
4),"V";
240 IF(PEEK(14400)AND128)=128ANDB=0ANDY>3.5THENGOSUB340
250 IFG=1THENGOSUB350
255 IFINT(Y)=1THEN270
260 GOTO210

```

```

48,48,48,112,240,240,0,12,12,12,14
,15,15,15,0
1100 GRAPHICS 3+16:FOR I=0 TO 2:SETCOL
OR I,0,0:NEXT I
1105 C=0:FOR I=0 TO 39:C=C+1:IF C=4 TH
EN C=1

```

Draws opening display on the screen.

```

1110 COLOR C:PLOT I,0:PLOT 39-I,21:NEX
T I
1120 C=0:FOR I=0 TO 21:C=C+1:IF C=4 TH
EN C=1
1130 COLOR C:PLOT 39,I:PLOT 0,21-I:NEX
T I
1140 COLOR 1:PLOT 9,4:DRAWTO 5,4:DRAW
0 5,6:DRAWTO 9,6:DRAWTO 9,9:DRAWTO 5,9
:PLOT 11,9:DRAWTO 11,4:DRAWTO 15,4
1142 DRAWTO 15,7:DRAWTO 11,7:PLOT 17,9
:DRAWTO 17,4:DRAWTO 21,4:DRAWTO 21,9:P
LOT 17,6:DRAWTO 21,6:PLOT 27,4
1144 DRAWTO 23,4:DRAWTO 23,9:DRAWTO 27
,9:PLOT 33,4:DRAWTO 29,4:DRAWTO 29,9:D
RAWTO 33,9:PLOT 29,6:DRAWTO 33,6
1150 PLOT 2,17:DRAWTO 2,12:DRAWTO 6,12
:DRAWTO 6,15:DRAWTO 2,15:PLOT 5,16:PL
OT 6,17:PLOT 12,12:DRAWTO 8,12
1152 DRAWTO 8,17:DRAWTO 12,17:PLOT 8,1
4:DRAWTO 12,14:PLOT 18,12:DRAWTO 14,12
:DRAWTO 14,14:DRAWTO 18,14
1154 DRAWTO 18,17:DRAWTO 14,17:PLOT 24
,12:DRAWTO 20,12:DRAWTO 20,17:DRAWTO 2
4,17:PLOT 26,12:DRAWTO 26,17
1156 DRAWTO 30,17:DRAWTO 30,12:PLOT 36
,12:DRAWTO 32,12:DRAWTO 32,17:DRAWTO 3
6,17:PLOT 32,14:DRAWTO 36,14
1160 RETURN

```



GALACTIC CHASE™

The aliens have swept undefeated across the galaxy. You are an enterprising star ship captain—the final defender of space.

As the aliens attack, you launch a deadly barrage of missiles. Flankers swoop down on your position. Maneuvering to avoid the counterattack, you disintegrate their ships with your magnetic repellers.

As your skill improves, the attackers increase their speed. And as a last resort, the aliens use their invisible ray to slow the speed of your missile launcher.

GALACTIC CHASE provides Atari owners with the most challenging one or two person game in the galaxy.



Atari 400/800 16k. Written in machine language. Requires joysticks.

Payment: Personal Checks—allow three weeks to clear.

American Express, Visa, & Master Charge—include all numbers on card. Please include phone number with all orders. 24.95 for cassette or 29.95 for disk plus 2.00 shipping. Michigan residents add 4%.

Check the dealer in your local galaxy. Dealer inquiries encouraged.

Galactic Chase © 1981 Stedek Software.



SPECTRUM
COMPUTERS

Dept S.
26618 Southfield
Lathrup Village, MI. 48076
(313) 559-5252

APPLE CAPTURE by William J. Ryan

TRS-80™ Version

1100 FORWTRD
1110 READLN
1120 WRTLN
1130 WRTLN
1140 WRTLN
1150 WRTLN
1160 WRTLN

Apple Capture is a HiRes color graphics game of deduction for two players, requiring AppleSoft and 32K RAM.

Your score is increased by one point for every apple you win. The game continues until you win 10 apples.

APPLESOFT MUSIC
"APPLE CAPTURE"
AUTHOR: WILLIAM J. RYAN
© 1982 SOFTSIDE

Main program:
10 GOSUB 20000
20 HND: PRINT "DO YOU WANT INS
30 TRAILONES": GOTO 40
40 IF AN = "Y": GOTO 50
50 LRI(1) = 1: LRI(2) = 1
60 P = 1: LRI(1) = 1: LRI(2) = 1
70 P = 2: LRI(1) = 1: LRI(2) = 1
80 P = 1: LRI(1) = 1: LRI(2) = 1
90 P = 2: LRI(1) = 1: LRI(2) = 1
100 P = 1: LRI(1) = 1: LRI(2) = 1
110 P = 2: LRI(1) = 1: LRI(2) = 1
120 GOSUB 2200
130 GOSUB 2200
140 GOSUB 2200
150 GOSUB 2200
160 GOSUB 2200
170 GOSUB 2200
180 GOSUB 2200
190 GOSUB 2200
200 GOSUB 2200
210 GOSUB 2200
220 GOSUB 2200
230 GOSUB 2200
240 GOSUB 2200
250 GOSUB 2200
260 GOSUB 2200
270 GOSUB 2200
280 GOSUB 2200
290 GOSUB 2200
300 GOSUB 2200
310 GOSUB 2200
320 GOSUB 2200
330 GOSUB 2200
340 GOSUB 2200
350 GOSUB 2200
360 GOSUB 2200
370 GOSUB 2200
380 GOSUB 2200
390 GOSUB 2200
400 GOSUB 2200
410 GOSUB 2200
420 GOSUB 2200
430 GOSUB 2200
440 GOSUB 2200
450 GOSUB 2200
460 GOSUB 2200
470 GOSUB 2200
480 GOSUB 2200
490 GOSUB 2200
500 GOSUB 2200
510 GOSUB 2200
520 GOSUB 2200
530 GOSUB 2200
540 GOSUB 2200
550 GOSUB 2200
560 GOSUB 2200
570 GOSUB 2200
580 GOSUB 2200
590 GOSUB 2200
600 GOSUB 2200
610 GOSUB 2200
620 GOSUB 2200
630 GOSUB 2200
640 GOSUB 2200
650 GOSUB 2200
660 GOSUB 2200
670 GOSUB 2200
680 GOSUB 2200
690 GOSUB 2200
700 GOSUB 2200
710 GOSUB 2200
720 GOSUB 2200
730 GOSUB 2200
740 GOSUB 2200
750 GOSUB 2200
760 GOSUB 2200
770 GOSUB 2200
780 GOSUB 2200
790 GOSUB 2200
800 GOSUB 2200
810 GOSUB 2200
820 GOSUB 2200
830 GOSUB 2200
840 GOSUB 2200
850 GOSUB 2200
860 GOSUB 2200
870 GOSUB 2200
880 GOSUB 2200
890 GOSUB 2200
900 GOSUB 2200
910 GOSUB 2200
920 GOSUB 2200
930 GOSUB 2200
940 GOSUB 2200
950 GOSUB 2200
960 GOSUB 2200
970 GOSUB 2200
980 GOSUB 2200
990 GOSUB 2200
1000 GOSUB 2200

SoftSide
January 1982 Vol. 5, No. 4 THREE DOLLARS

Off and running with GAMBLER

SoftSide
November 1981 Vol. 5, No. 2
THREE DOLLARS

Special Issue: Music in the Micro

Flight of the Bumblebee

SoftSide
December 1981 Vol. 5, No. 3 THREE DOLLARS

TITAN

SoftSide
Your BASIC Software Magazine • NUMBER ELEVEN
AUGUST 1981

Battle
Sharl
Tom

Back Issues
from
SoftSide™

Check for successful docking with mother ship.

```
270 IFX<>S+5THEN500
275 P2=P2+1
280 FORI=2T015:PRINT@I#64+3,STRING$(57,32);:NEXT
```

If all people are gone from the bottom of the screen, then increase difficulty level and branch to bonus routine.

```
285 IFP1=0THENP1=6:D1=D1+1:GOTO650
290 GOTO30
```

Subroutine to move ship according to input.

```
300 P=PEEK(14400)AND98:IFP=32ANDX>6THENX1=X-2
305 IFP=34ANDX>5THENX1=X-1
310 IFP=64ANDX<57THENX1=X+2
315 IFP=66ANDX<58THENX1=X+1
320 RETURN
```

Launch missile.

```
340 PRINT@0,;:FORAZ=10T0100STEP10:SOUNDAZ,3:NEXT:G=1:XM=X:YM=Y
```

Move missile and check for collision with meteors.

```
350 PRINT@XM+(INT(YM+.5)*64), " ";YM=YM-1
360 A=PEEK(15360+XM+(INT(YM+.5)*64)):IFA=140THENPRINT@XM+(INT(YM+.5)*64), " ";G=0:SC=SC+20:PRINT@959+L,USINGU$;SC/1E5;:RETURN
370 PRINT@XM+(INT(YM+.5)*64), " ! ";:IFYM<3THENPRINT@XM+(INT(YM+.5)*64), " ";G=0
380 RETURN
```

Successful landing routine. Wave arms of the next person and move him over to rocket.

```
400 Y=19-P1:X=62:X1=58:X2=L+8:IFP1<4THENY=16-P1:X=1:X1=2:X2=L-4
410 FORI=1T04:FORJ=43T088STEP45:PRINT@Y#64+X,CHR$(J);:FORW=1T030:NEXTW,J
420 PRINT@Y#64+X1,"YAY!";:IFI/2=INT(I/2)THENPRINT@Y#64+X1," "
;:PRINT@0,;:FORAZ=1T02:FORAY=77T07STEP-7:SOUNDAZ,4:NEXT:FORAY=7T077STEP7:SOUNDAZ,4:NEXT:NEXT
430 NEXTI:PRINT@Y#64+X, " ";
440 FORI=X+SGN(X1-X) TO X2 STEP SGN(X2-X)
450 FORJ=43T088STEP45:PRINT@960+I,CHR$(J);:FORW=1T020:NEXTW
455 PRINT@0,;:SOUND100,0:SOUND200,0:NEXTJ:PRINT@960+I, " ";:NEXTI
460 X=I+SGN(X1-X2):Y=15:FORI=1T02:X=X+SGN(L-X1):Y=Y-1
465 FORJ=43T088STEP45:PRINT@Y#64+X,CHR$(J);:FORW=1T020:NEXTW:PRINT@Y#64+X, " ";:NEXTI
470 P1=P1-1:U=1:GOTO200
```

Explosion routine.

```
500 REM
520 X1=X-2:X2=X+2:X0=X:YS=2:PRINT@X-1+(INT(Y+.5)*64), " ";:PRINT@X+(INT(Y+1.5)*64), " ";
530 FORI=YT014:PRINT@X2+I#64,CHR$(170)CHR$(173);:PRINT@X1+I#64,CHR$(158)CHR$(149);:IFU=1THENPRINT@X0+I#64,"X";
540 PRINT@0,;:FORAZ=1T010:SOUNDRND(255),RND(2)-1:NEXT:YS=YS-.25:IFYS<0THENYS=0
550 PRINT@X1+I#64, " ";:PRINT@X2+I#64, " ";:IFU=1THENPRINT@X0+I#64, " ";
560 X1=X1-YS:IFX1<5THENX1=58
```

```
570 X2=X2+YS:IFX2>58THENX2=5
580 NEXTI:SL=SL-1:IFSL=0THEN600
590 GOTO280
```

End of game. Print score and game-over message and wait for input.

```
600 CLS:PRINT@409,"* GAME OVER *";:PRINT@533,"YOUR SCORE IS ";USINGU$;SC/1E5
610 PRINT@658,"PRESS <ENTER> TO PLAY AGAIN";
620 IFINKEY#<>CHR$(13)THEN620ELSEGOTO15
```

Bonus routine. Awards 50 points for each person safely brought to mother ship.

```
650 CLS
660 PRINT@411,"* BONUS *":PRINT@536,"SCORE = ";USINGU$;SC/1E5
670 FORI=1T0P2:PRINT@664+I#2,"X";:SC=SC+50:PRINT@544,USINGU$;SC/1E5
680 PRINT@0,;:FORAZ=1T010:SOUNDI#15+30,3:SOUNDI#15+60,3:NEXT:FORW=1T050:NEXTW:NEXTI
690 PRINT@0,;:FORAZ=1T0100:SOUND100-AZ,2:SOUNDAZ,2:NEXT:IFP2=6THENPRINT@406,"*** BONUS SHIP! ***";:SL=SL+1:FORP2=1T04:FORAZ=80T0185STEP7:SOUNDAZ,1:SOUNDAZ+50,1:NEXT:NEXT
```

Bonus ship routine. Awards a bonus ship if all six people are rescued.

```
695 FORW=1T011STEP3:FORAZ=WTOW-7STEP-1:SOUNDAZ,2:NEXT:NEXT
700 GOTO20
```

Initialize sound routine.

```
1000 Z=0:FORX=1T0158:READY:Z=Z+Y:NEXT:IFZ<>15204THENCLS:PRINT"DATA BASE ERROR IN LINES 60060-60160, CHECK LISTING.":PRINT:LIST 1030-1080ELSEY=86:X=255:POKE-1,0:IFPEEK(-1)<>0THENX=192:POKE-16385,0:IFPEEK(-16385)<>0THENX=127
1005 POKE 16562,X:POKE 16561,Y:CLEAR500:A1=PEEK(16561)+2:A2=PEEK(16562):A=A1+A2#256:Z=A-1:FORX=1T0158:Z=Z+1:Z=Z+65536*(Z>32767)
1010 READY:IFY<0THENY=A1+ABS(Y):POKEZ,Y+256*(Y>255):Z=Z+1:POKEZ,A2-(Y>255):NEXTELSEPOKEZ,Y:NEXT
1015 IFPEEK(16396)=201POKE16526,A1:POKE16527,A2ELSECMD"":DEFUSR=A1+(A2+256*(A2>127))#256:POKE14308,0
1020 IFPEEK(16807)+PEEK(16808)#256<>A+24THENA=USR(0)
1025 GOTO15
1030 DATA58,166,65,50,-164,42,167,65,34,-165,62,195,50
1035 DATA166,65,33,-24,34,167,65,201,245,123,254,2,40,4,254
1040 DATA16,32,79,229,213,42,230,64,126,183,32,4,35,35,35,35
1045 DATA215,6,5,17,-156,26,190,32,104,19,35,16,248,43,215
1050 DATA43,34,230,64,241,241,241,197,213,215,205,55,35
1055 DATA229,205,127,10,42,33,65,34,-167,225,215,43,34,230,64
1060 DATA35,205,55,35,43,229,205,127,10,42,33,65,58,-167,60
1065 DATA183,87,24,4,24,48,24,44,66,62,1,211,255,16,252,66,62
1070 DATA2,211,255,16,252,58,64,56,230,4,32,7,124,181,40,3,43
1075 DATA24,228,175,50,154,64,225,209,193,215,195,30,29,83,79
1080 DATA85,78,68,209,225,241
```

Draws opening display on the screen.

```
1100 CLS:PRINTCHR$(23);
1110 PRINT@208,CHR$(191)STRING$(13,143)CHR$(191):PRINT@272,CHR$(191)" S P A C E "CHR$(191):PRINT@336,CHR$(191)" R E S C U E "CHR$(191):PRINT@400,CHR$(191)STRING$(13,188)CHR$(191)
1120 PRINT@648,"ORIGINAL BY MATT RUTTER":PRINT@716,"S-80 BY ALAN J ZETT"
1130 GOTO1000
```

Let's Be Civilized

by Leonard Buchanan

There are times when something you have known all along suddenly dawns on you anew. You view the world in a different perspective: things which were commonplace are no longer, and it is as though you were a child again, seeing through purer eyes. Such a thing happened to me not long ago, and the experience is one which I'm realizing should be shared with my fellow computer addicts.

It all goes back to a time when I was a child and we were visiting at the home of a family friend. The proud parents were bubbling with enthusiasm over their little Johnny, who had just learned how to do some dumb little thing. When it was my turn to show approval, I said, "SO?"

Shortly thereafter, it was very indelibly explained to me that this is not the civilized way to treat people. You are nice to people. You are nice to their little Johnnies. You are nice to their dog, if need be, but above all, be nice! That is the "civilized" way to live.

This lesson had stuck with me for years. I have oohed and aahed over many a goo-goo and gaa-gaa, and been quite "civilized," actually, but somewhere in the flow of many years rushing by, the lesson was lost, and, when it was least expected, the "uncivilized" beast in me arose.

A friend of mine was showing off his H89 computer, and all the "really neat things" it would do. I, an avid ATARI® freak, was duly unimpressed, and when he turned for approval, I muttered an unapproving "nice."

He threw the gauntlet back into my face with, "Yeah! A lot better than that piece of junk you work on!"

Well! Listen, now! You can beat me severely about the head and shoulders with a big stick. You can cheat me out of money or fame. You can dent the fenders on my old car (not that anyone would notice, anyway), but **YOU DON'T CALL MY ATARI® A PIECE OF JUNK!**

From that point, it was toe to toe, and nose to nose. He was screaming idiotic things like "built-in monitor" and "14 billionth of a nanosecond," to which I was screaming back really intelligent things like "So? So?"

One of the reasons God gave men wives, was to break up this sort of thing. Our wives pulled us apart just in time, for I was about to get down and dirty, and hit him with the final death blow: "Color graphics!! Color graphics!!"

Looking back on it now, from the quiet of my keyboard, I see that what happened the other day is no different from those many years ago at little Johnny's. My friend wasn't showing off a machine, he was showing off his KID.

The computer which you have in your home quickly becomes a part of the family. Between the first day when you bring it home, all wrapped up, and treat it as though the slightest little thing could ruin it forever, to the day when someone sits at it, afraid to touch it, and you say reassuringly, "That's OK — there's really no way to hurt it at all" there are many times when you show off its new little abilities, have your picture taken with it, and are quietly bursting with pride over the new things you have "taught" it to do. It follows in your own footsteps, however, you don't want it to make the same mistakes you've made.

Many people name their computers and why not, really? Our cars have names. My wife even has a few choice names for her electric can opener (none of which can be printed in this magazine). If they have names, why not our computers?

But, that's off the subject. The whole purpose of this was to remind you that little machine sitting there in my living room is like a kid to me. **DON'T CALL IT JUNK!**

And from now on, I'll try to be more "civilized" about that brat of yours, too.



THE *SOFTSIDE* 1982 READER SURVEY

SoftSide takes pride in striving to address both you and your computer in your respective languages. In order to improve our services and plan for the future of the magazine, we need to know as much as possible about you and your computer system.

We are publishing this rather extensive survey to gather information we need to know in order to continue to bring you the finest software publication on the market today. Please take the time to carefully fill it out and return it to us. The information that you will provide will bring its own reward: a better *SoftSide* and special attention for YOU and your computer.

1. In order to ensure that you will continue to receive as much, if not more information as you have in the past, we may soon be forced to publish computer-specific editions to accommodate additional computers. In order to even consider such a move, we absolutely must know which computer you own. On the second line of the address label you should find a three digit code designating the computer and version of *SoftSide* to which you subscribe. Here is a key to that code:

TRM — TRS-80® MAGAZINE ONLY
TRC — TRS-80® CASSETTE
TRD — TRS-80® DISK
APM — APPLE™ MAGAZINE ONLY
APC — APPLE™ CASSETTE
APD — APPLE™ DISK
ATM — ATARI® MAGAZINE ONLY
ATC — ATARI® CASSETTE
ATD — ATARI® DISK

If there is no code, we do not have you designated for a specific computer. If there is any wrong information on your address label, especially in the code designating your computer, please attach the label in the space provided below and make any corrections necessary in the address box following.

Name _____

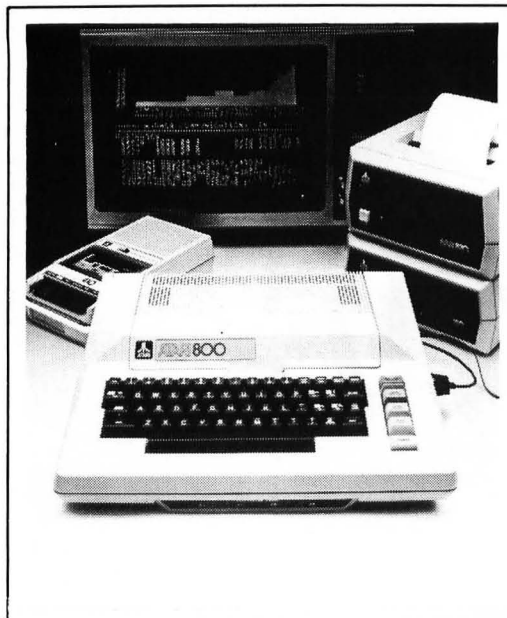
Address _____

City _____ State _____ Zip _____

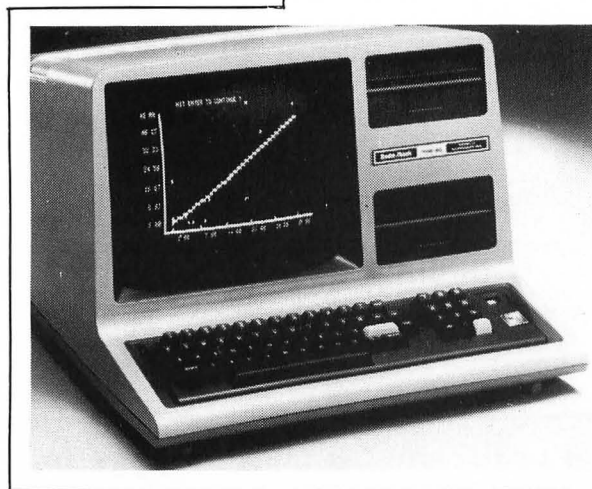
2. I own an:

- APPLE™ II
- APPLE™ II PLUS
- APPLE™ III
- ATARI® 400
- ATARI® 800
- TRS-80® MOD I
- TRS-80® MOD II
- TRS-80® MOD III
- TRS-80® Color Computer
- TRS-80® Pocket Computer
- Other _____

ATARI®



System Specific



TRS-80®



APPLE™

-
1. Memory Size _____
 2. Keyboard Yes No
 3. Keypad Yes No
 4. Lowercase Modification Yes No
 5. Printer Graphics Text only Col width _____
 6. Monitor Color B/W
 7. Cassette Yes No
 8. Disk Drive(s) No. of Drives _____
 Single density Double density
 9. Disk Operating System
 TRSDOS 2.3 LDOS
 NEWDOS 80 APPLE 3.3
 DOSPLUS APPLE 3.2
 NEWDOS+ ATARI 1.0
 VTOS ATARI 2.5
 10. Languages BASIC Pascal APL
 Fortran Cobol Pilot Other _____
 12. Interface (Brand name & Model) _____
 13. Light Pen Yes No
 14. Paddles Yes No
 15. Joysticks Yes No
 16. Graphics tabloid Yes No
 17. Modem Yes No
 18. Plotter Yes No
 19. Three favorite software programs or pkgs
A. _____
B. _____
C. _____
-



Get
on
with
the
fun!

APPLE™
ATARI®
TRS-80®



Disk Version

SoftSide's Disk Version for the Apple™, ATARI® or TRS-80® is today's best investment in computer software. Subscribers receive a minimum of four quality programs and databases per month, at an average cost of \$2.60 each. In 1981, **SoftSide** DV published "NEWBASIC," "Envyrn," "National Anthems," "Volleyball," "Mean Checkers Machine," and "Bobsledding" in addition to all the programs listed in **SoftSide**.

SoftSide DV will allow you to enjoy your system and your programs without the tiresome typing of line listings AND you won't waste additional time hunting for your typing errors. Simply insert the disk, boot, and you'll be ready to type COMMANDS the day you get your diskette.

A subscription to **SoftSide** DV is more than a savings — it's an investment in an ongoing library of software — multiple and Machine Language programs which would be difficult to read and even more difficult to key into your system. You'll spend your software dollars wisely with a subscription. Where else could you find four programs on disk AND a software magazine, for less than \$10.50?

You Can Do All This For Less

SoftSide DV, including twelve disks and magazines, costs only \$125 per year. But, we'll make it easy on your budget: We'll bill you only \$32.50 per month for four months. (You probably spend more than that for most of the programs you buy now.) The installment plan includes \$5 to cover our extra billing costs. A subscription to **SoftSide** DV gives you a minimum of 48 quality programs and twelve issues of **SoftSide** per year — a true value.

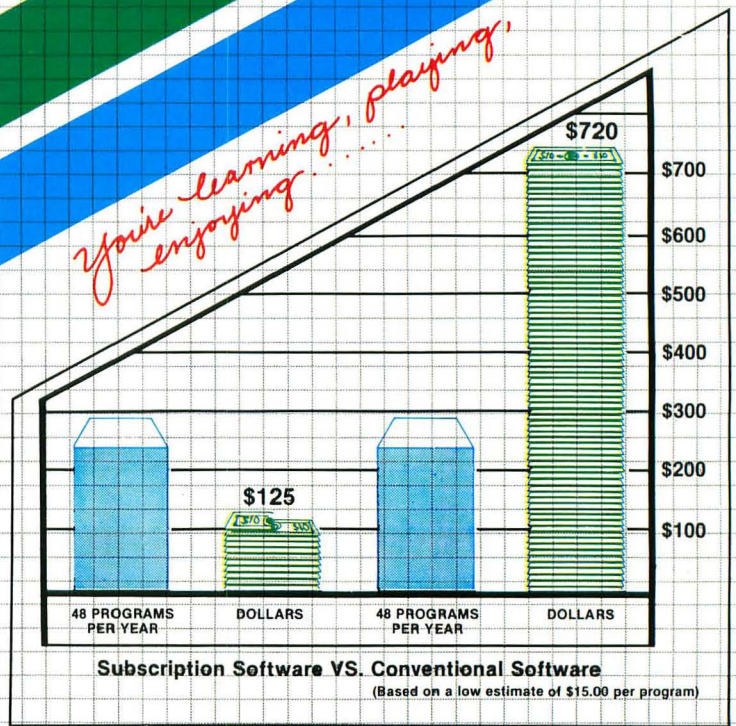
Order your **SoftSide** DV subscription TODAY. Use the coupon below or the bind-in card elsewhere in this issue.

You've booted up - start your first program

You're learning, playing, enjoying

Avoid the Toil and Get On With the Fun — DV Is a Smart Investment

OOPS! Support!

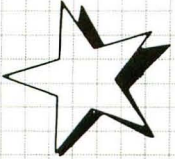


DATA

TIME

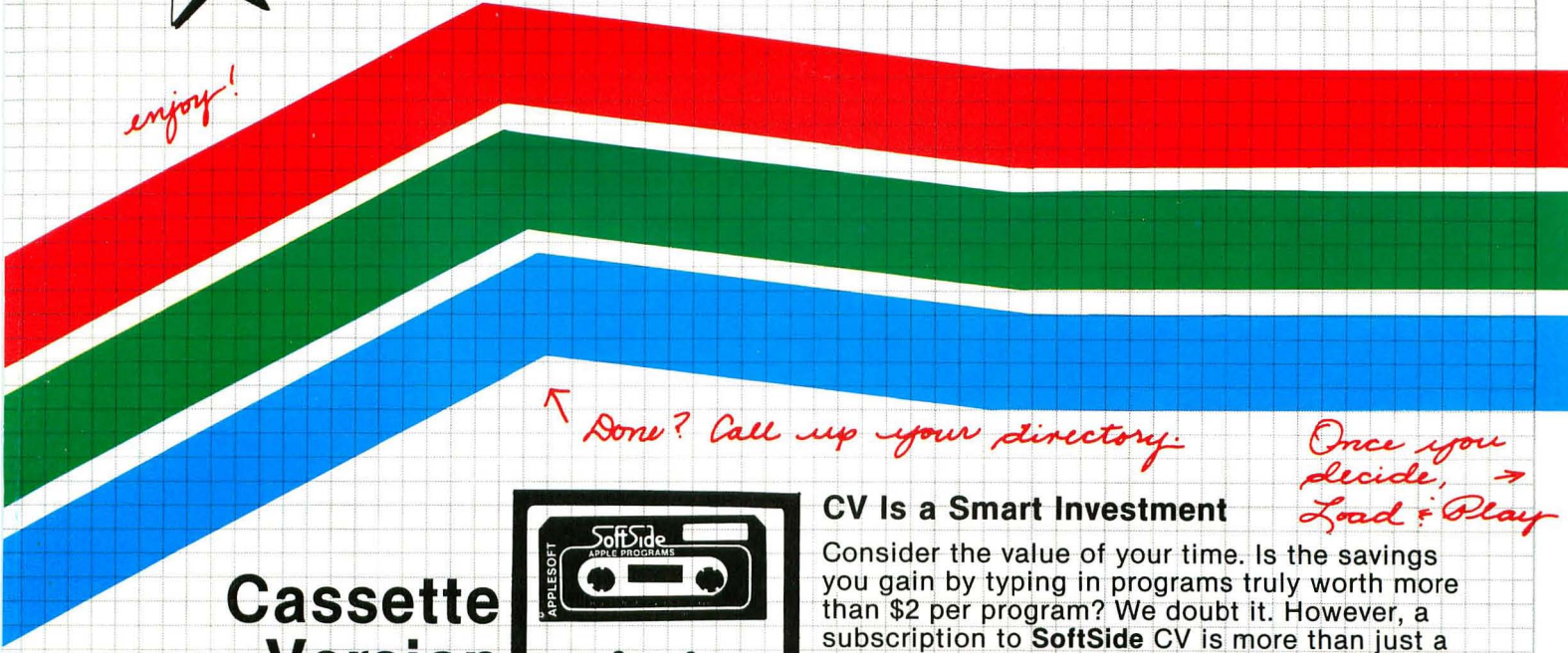
Extended to March 15th!!

BONUS — ~~Subscribe before January 31, 1982~~



And receive a disk or cassette containing some of our finest software from the past. Disk to include all of our 1981 enhancements!

enjoy!



Done? Call up your directory.

Once you decide, → Load & Play

Cassette Version



The **SoftSide** Cassette Version (CV) is a very economical and time-saving investment in computer software for you and your Apple™, ATARI® or TRS-80®. You'll get all of the programs for your system published in **SoftSide** on cassette every month at an approximate cost of \$2 per program.

SoftSide CV lets you enjoy your computer and programs without the tiresome typing of line listings, AND you won't waste additional time hunting for your typing errors. Simply load the cassette and you're ready to run.

CV Is a Smart Investment

Consider the value of your time. Is the savings you gain by typing in programs truly worth more than \$2 per program? We doubt it. However, a subscription to **SoftSide** CV is more than just a savings — it's an investment in an ongoing library of software. Where else can you get three programs on cassette AND a software magazine for less than \$6.25 per month?

A subscription to **SoftSide** CV, including twelve cassettes and magazines, costs only \$75 per year. You probably bought your cassette system because it was an economical way to enter computing. Now save even more money with a subscription to **SoftSide** CV.

Stop typing and order your CV subscription TODAY! Use the coupon below or the bind-in card located elsewhere in this issue. You'll be enjoying the value of **SoftSide** CV in only a few weeks.

SUBSCRIBE TO DV OR CV NOW AND RECEIVE YOUR BONUS GIFT

With **SoftSide** CV or DV you'll get each month's Apple™ ATARI® or TRS-80® programs delivered on tape or disk. All programs are tested and ready to run.

- SoftSide** Disk Version \$125/year **SoftSide** Cassette Version \$75/year
- Four monthly payments of \$32.50 (includes \$5 billing charge)

Which computer? Apple™, TRS-80®, ATARI®

Check here if you would like to have the remainder of your current **SoftSide** subscription converted at the rate of \$4.25/cassette or \$8.42/disk.

Name _____

Address _____

City/State _____ Zip _____

MasterCard VISA Name of Cardholder _____

MC# and Interbank#/VISA# _____

Exp. Date _____ Signature _____



Penn. 1-800-662-2444
1-800-345-8112

Mail to:

**SoftSide Publications
Media Version
515 Abbot Drive
Broomall, PA 19008**

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.

Another Typo!

Still Typing....



**Stop
typing
line listings...**

**Start
typing
commands!**

General

3. What is your sex?
 Male Female
4. What is your age?
 under 12 13-18 19-24 25-29
 30-35 36-40 41-50
 50 and Over
5. Highest level of Education
 Grades 1-8
 Grades 9-11
 Graduated High School
 Technical or Trade School
 Some College
 Graduated College (4 year)
 Some postgraduate work
 Postgraduate degree
6. How many family members make use of your computer? _____
7. What are the ages of other family members using your computer? (Check more than one if applicable)
 under 12 13-18 19-24 25-29
 30-35 36-40 41-50
 50 and Over
8. How many people outside your family use your computer? _____
9. With which of the following hobbies would you identify yourself?
 Amateur Radio
 Fantasy Role Playing Games
 Video
 Writing
 Other _____
10. What was your family's total income in 1981 (before taxes)?
 Less than \$5000
 \$5000 to \$7499
 \$7500 to \$9999
 \$10,000 to \$14,999
 \$15,000 to \$19,999
 \$20,000 to \$24,999
 \$25,000 to \$29,999
 \$30,000 to \$39,999
 \$40,000 to \$49,999
 \$50,000 to \$74,999
 Over \$75,000
11. Which of the following categories best describes your occupation? (Please check only one)
 Professional Technical Educational
 Management Sales Clerical
 Craftsman Laborer Homemaker
 Farmer Military Unemployed
 Student Retired Other _____
12. Are you self-employed? Yes No
13. How often do you use computers in your business?
 Often Sometimes Never
14. Do you use your home computer for business?
 Often Sometimes Never
15. Would you consider yourself a "programmer"?
 Yes No
16. What was/is your reason for learning to program?
 Career
 Self-Improvement
 Entertainment
 Preparation for the future.
17. Is your computer in a room by itself?
 Yes No
18. Have any of your friends bought a computer based on your advice?
 Yes No
How many? _____
19. Do you belong to a computer club or users' group?
 Yes No
20. Do you subscribe to a movie channel?
 Yes No Not Available
21. Do you subscribe to cable TV?
 Yes No Not Available
22. Which of the following appliances do you own?
 Fire Alarm System
 Food Processor
 Microwave Oven
 Remote Control System (BSR)
 State-of-the-art Stereo (over \$1000)
 Telephone Answering Machine
 VCR
 Video Disc
 Video Game Machine (Other than your computer)
 Wide-Screen TV
23. Are you a subscriber to *SoftSide*?
 Yes No
24. If no, where do you obtain your copy?
 Newstand or Store _____
 Library School Friend Other
25. How did you discover *SoftSide*?
 Advertising in another magazine From a friend
 Dealer Gift Subscription Computer Club
 Library Direct Mail Other
26. If you do not currently subscribe to the cassette or disk version of *SoftSide*, why not?
 Cost Not that interested
 Do not own the necessary equipment
 Did not know about it Other
27. Please rate the following features of *SoftSide* in the order of their importance to you 1-9
____ Programs
____ Programming Articles
____ Entertainment Articles
____ Series
____ Columns
____ Reviews
____ Advertising
____ Hints and Enhancements
____ One-Liners/K-Byters

28. These seven feature programs appeared in *SoftSide* in past months. Please indicate.

1. typed complete program 2. partially typed program
3. intend to type program
4. do not intend to type program
- Chemistry Drill/July
 - Quest/August
 - Flip It/September
 - Leyte/October
 - Flight of the BumbleBee/November
 - Titan/December
 - Gambler/January

29. What other magazines do you subscribe to?

- Analog (ATARI®)
- Boston Computer Society Update
- Byte
- Classroom Computer News
- Compute
- Creative Computing
- Desktop Computing
- Infoworld
- Interface Age
- Micro the 6502/6809 Journal
- Microcomputing
- 80 Microcomputing
- Nibble
- Omni
- Personal Computing
- Popular Computing
- Popular Electronics
- Softalk
- Strategy & Tactics
- The Alternate Source
- The ATARI® Connection
- The Dragon
- The Space Gamer
- TRS-80® Newsletter
- 80 US
- Other _____

30. Do you subscribe to:

- Micronet?
- Telenet?
- The Source?

31. On an average, how much time do you spend on-line per month? _____

32. What is your average monthly telephone bill?

- Under \$50
- \$50-\$75
- \$75-\$100
- \$100-\$150
- Over \$150

33. From which of the following sources have you purchased the most computer software?

- Computer Stores
- Discount, Department or General Electronic Stores
- Magazine Advertisements (mail-order)
- Catalogs (mail-order)
- Telephone
- Other _____

34. On the average, how much do you spend yearly on commercial software?

- under \$20 \$20 to \$50
- \$50 to \$100 Over \$100

35. What kinds of programs do you prefer?

- Business Recreational Educational
- Utility Personal/Home Management
- Communications Other _____

36. Do you own a word processing program other than Microtext?

- Yes No

It's name(s) _____

37. Do you have a database manager other than the *SoftSide* Developing Database?

- Yes No

It's name(s) _____

38. Does your computer have its own monitor or television set? _____

39. How long have you owned the majority of your system? _____

40. Estimated total investment for this system? _____

Comments: _____

Mail to:
SoftSide Publications
6 South St.
Milford, NH 03055

SECTIONALIZATION INTRODUCTION

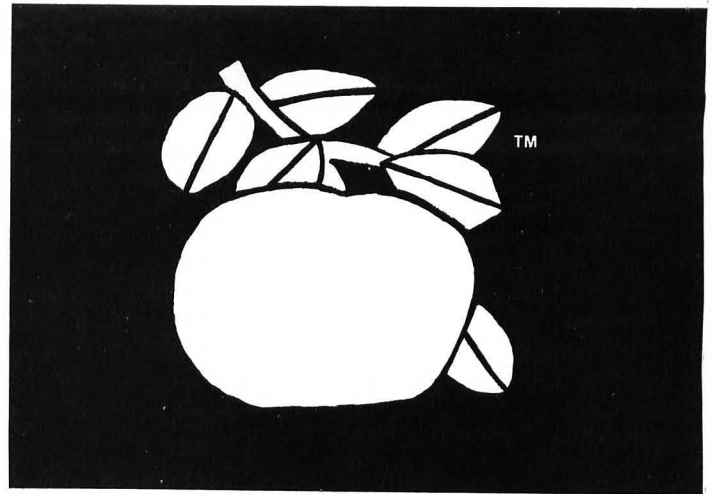
Welcome to the New "Sides" of *SoftSide*

This month we initiate a new format in *SoftSide*. In an attempt to make each issue easier for your use, we are incorporating articles and programs relating to specific computers into their own "side" of *SoftSide*. Programs and articles pertaining to more than one of the systems we support will be printed in the front of the magazine.

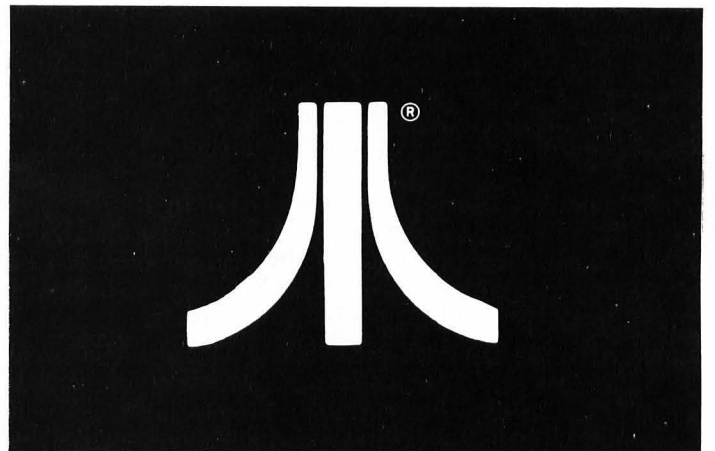
It is important that you, as a computerist, are continually aware of the entire realm of computing. We encourage you to read all of the "sides" of *SoftSide*, whether they specifically pertain to your computer or not. It is only through this vital exchange of information that we can all contribute to the continual growth of the art of computing.

We hope you enjoy the new format.

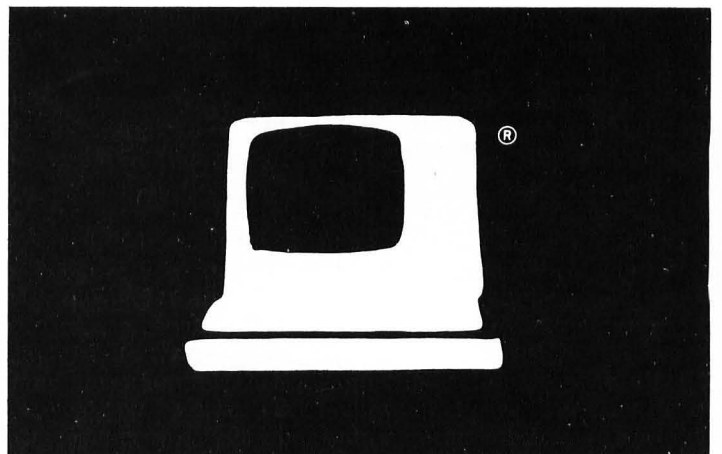
Our goal is to make *SoftSide* as user-friendly as possible and we will continue to refine the organization of the magazine over the next few months. Let us know your reaction. It's only through your input that we know the effectiveness of our efforts.



APPLE™/SIDE **44**
page_____



ATARI®/SIDE **59**
page_____



TRS-80®/SIDE **71**
page_____

ANDORRA

by Brent Packer

***Andorra* is a socio-political-economic simulation for an Apple II with Applesoft, 48K RAM, and disk drive. It is included as a bonus program on this month's Apple Disk Version.**

You have been chosen! You are the elected leader of Andorra. Don't worry too much about being honest — in fact, the people of your country don't much care what deals you have cooking under the table. Go ahead: Buy off members of parliament, amass your own personal fortune, even take a few bribes yourself. Your country is located between two natural enemies, France and Spain, so the selling of spying services could be quite lucrative.

The people certainly are understanding about these things that a politician **MUST** do. But you are still subject, as an elected official, to their whims. You have to pay off the people in order to stay in office. Their deal is simple: Spend, spend, spend, for everything that they want; and keep taxes reasonable.

With this agreement, the citizens (and you) would be mutually enriched, and quite happy. But even in Andorra, the laws of economics

cannot be suspended. Since tax money must finance expenditures, spending must be limited in order to have limited taxation. The agreement is thus made harder to carry out, but not impossible. Your job, as Governor, is to find the happy and profitable medium where the people get enough of what they want to keep you in office doing what you want.

Many features of *Andorra* are not immediately obvious just by casually running the program. The following run-through will help to introduce it.

After the initial instructions, which summarize the goals of the game, you are asked to take the oath of office. The people are very serious about this, so be sure you say "yes" to the oath — or watch out!

Serious business begins with the setting of the tax rates. You will set the annual percentage that the government will take from income in each category. The computer will then tell you how the people have reacted to your tax rates. A 39% private income tax, for instance, will cause reported income to drop a substantial amount (from people dodging high taxes). Because this is too high, a certain number of people will now vote against you in the next election, based on just exactly what tax rate you imposed. This tax will produce a lot of revenue the first

year, but reported income will drop, causing less tax revenue, and your popularity will drop. None of this is random, but is based entirely on economic principles. The computer will tell you the effects on property values, profits, reported income, sales volume, and foreign trade, caused by the five taxes you set (property, corporate income, private income, sales, and tariff).

After you set taxes, the parliament will present you with six spending bills it has passed. If you are conscious that you are spending too much, you can veto a bill to try to make parliament spend less. On the other hand, if you are vote-conscious you can veto a bill so that you can spend **MORE**. After you have attempted a veto, parliament will try to override you. This requires a two-thirds majority and, if you don't veto too much, they will go along with you more than two-thirds of the time. If your veto is sustained, then parliament rewrites the bill with a bias toward spending less or more.

Occasionally a surprise expense comes up (flood damage, etc.). If you can meet that expense, the voters reward you well. Occasional-



ly, also, you may be offered a bribe. Your chance of being caught is given, so that you know what you're getting into. In spite of what was said above, some people do NOT like it if they find out you've taken bribes. But then, it takes a lot more than being caught on the take to lose an election. Bribes are a good way to add to your personal fortune; and if you get into a jam — impeachment, a close election, etc. — you have more "power of persuasion."

At the end of each year of your three-year term you will see your balance sheet. It displays profit/loss, land holdings, inflation rate (determined by tax rates and spending), debt (with interest), and land values. If you have a loss, you must make it up by selling land, borrowing from the bank, or printing more money. Selling land costs a few votes — remember that the people want MORE. Borrowing costs no votes but does cost an increasing amount of interest and may lead to being denied further loans by the


banks. Printing money doesn't cost votes either, but the resulting inflation does. If you have a profit, it is automatically put into the bank and draws an increasing amount of interest as your deposits increase.

After you're all finished accounting for the year, the opinion poll department takes a poll of your popularity. It tells any ways that you might have lost votes, so that you can try to do better before the election. Then it's off into another year.

When your term is up, you can hire an ad agency to run your campaign. This is usually a good investment, as their commercials can pick up a few last-minute votes.

Andorra has the electoral college system, which sometimes gives you an advantage and sometimes a disadvantage. If a third party appears, and draws enough votes from

either your party or the other one, then the parliament decides the election, with each Andorran state having one vote. It can take ten or twelve ballots before the election is decided. Since no one wants to lose after coming so close, this part of the election (if it is triggered) is biased somewhat in your favor. It's great fun to watch the balloting, but you still can lose quite often.

If you lose, you are told why. But if you win, then you have great cause to rejoice: You have found out how to fulfill your original agreement with the people. You then have proved your competence to rule a real country. This could mark the beginning of your fabulous political career! 



K-Byter

Pick-Up

An Applesoft K-Byter by William Pu, Albany, NY

This program displays a Hi-Res field of hollow and solid blocks, enclosed by a border. The object of the game is to "pick up" all of the hollow blocks displayed on the screen, without hitting any of the solid blocks or the border. You start in the center of the screen, and control your "man" using the usual screen-editing keys: I, J, L, and M for up, left, right, and down. As your man moves, he leaves a solid trail behind him, which also must be avoided: Crossing or turning back on your trail will end the game.

There are 30 (!) skill levels available, ranging from the fairly simple to the virtually impossible. If you do win by picking up all the hollow blocks, you are given a 100% rating. If the game ends before you have won, you are given a lower rating based on the number of blocks you have managed to pick up.

```
0 H = 4:I1 = 6:J = 1:K = 16384:L =
  70:M = 200:N = 204:O = 206:P
  = 200: GOSUB 2000
2 TEXT : PRINT :Z1 = 4: POKE -
  16368,0: INPUT "LEVEL?(1-30)
  ";LE:W = LE * 2: GOTO 700
100 Y = Y - J: RETURN
110 X = X - J: RETURN
120 X = X + J: RETURN
130 POP : GOTO 150
140 Y = Y + J: RETURN
150 FOR I = J TO 60 - W * 2 STEP
  I1:Z = PEEK ( - 16384): NEXT
  : IF Z > M AND Z < O AND Z <
  > N THEN Z1 = Z - P
```

```
182 ON Z1 GOSUB 100,110,120,130,
  140: GOSUB 4020: GOSUB 4000
186 IF G = 2 THEN HT = HT + J
188 IF HT = W OR G = 1 THEN 3000
190 GOSUB 4010: GOTO 150
700 HGR2 : HGR : POKE - 16302,0
  : SCALE= 1: ROT= 16: HCOLOR=
  7:X = 35:Y = 16: GOSUB 4010:
  FOR I = J TO W
711 GOSUB 800: IF G < > 0 THEN
  711
725 GOSUB 4010
730 GOSUB 800: IF G < > 0 THEN
  730
765 POKE X * L + Y + K,2: XDRAW
  2 AT X * H,Y * I1: NEXT : HPL0T
  0,0 TO 0,191 TO 279,191 TO 2
  79,0 TO 0,0:X = 35:Y = 16: GOTO
  150
800 Y = INT ( RND (1) * 32):X =
  INT ( RND (1) * 69): GOSUB
  4000: RETURN
2000 FOR I = 7676 TO 7703: READ
  A: POKE I,A: NEXT : POKE 232
  ,252: POKE 233,29: RETURN : DATA
  2,0,6,0,19,0,45,45,37,36,63,
  63,55,46,45,37,63,63,0,45,45
  ,37,36,63,63,55,14,0
3000 TEXT : PRINT HT / W * 100%
  ": RUN
4000 G = PEEK (X * L + Y + K): RETURN
4010 DRAW J AT X * H,Y * I1: POKE
  X * L + Y + K,J: RETURN
4020 IF Y < 0 OR Y > 31 OR X < 0
  OR X > 69 THEN 3000
4030 RETURN
5000 REM WILLIAM PU
5001 REM 19 DELAFIELD DRIVE
5002 REM ALBANY, NY 12205
5003 REM 11/23/81
```

5

RUBIK - CUBE



by F.J. Condo, Jr.

RubiCube is a color-graphics simulation for a 16K Apple with Applesoft in ROM.

Accompanied by cover features in such prestigious magazines as *Scientific American* and *Time*, the *Rubik's Cube* puzzle has become a national phenomenon in recent months. The ingenious device is a cube which is subdivided into a number of smaller cubelets, attached together in such a way that they can be rotated in three dimensions. On each face of the larger cube, nine faces of the smaller cubelets are visible. Each face of each cubelet is one of six colors; and when these are all oriented so that the larger cube has one solid color on each of its six faces, the cube is said to be "solved." This program allows you to display and manipulate the *Rubik's Cube* with your computer.

In order to represent the three-dimensional cube on the flat screen of a TV or monitor, it must be visualized as a cardboard box that's been cut along the edges and laid out flat. That is how the cube appears on the screen in low-resolution

graphics. Each face of the cube has a one-letter name that is the first letter of its position. The names are T)op, B)ottom, L)eft, R)ight, F)ront, and P)osterior.

You can rotate each face a quarter turn clockwise or counterclockwise, or a half-turn (which is the same in either direction). The basic command is a face name followed immediately by a directional symbol. The directional symbols are + for clockwise, - for counterclockwise, and 2 for a half turn (180 degrees about). Thus some basic commands are:

F+	Front face clockwise
P-	Posterior face counterclockwise
L2	Left face 180 degrees about

You may also string basic commands together for uninterrupted processing. So you could type this:

T2B2L2R2P2F2

which would produce a checkerboard pattern from the solved cube. If you should make a typographical error somewhere in a long command

string, the cube will be restored to the state it was in before you typed the erroneous string. Don't include any spaces or other extraneous characters.

On the screen, the P face is the one far to the right. (Remember the cut-open cardboard box.) The F face is at the center of the cross formed by T, R, B, and L. The condensed instructions at the bottom of the screen will remind you of this.

To spin the whole cube, use the S command. Unlike the basic commands, the S command is typed alone on a line. You will then be prompted for the axis about which you want to spin the cube, and then the direction (+, -, or 2). The X axis goes through the L and R faces; the Y axis goes through the T and B faces; and the Z axis goes through the F and P faces. The directions are keyed to the direction that the R, T, and F faces rotate, respectively. Thus, the series of commands S, Z, -, would rotate the cube in such a way as to make the F face move in a counterclockwise direction.

The D command stands for disk/tape operations. It is used for saving to and recalling from disk or tape.

This allows you to save a cube in progress, or to save one before you attempt a tricky sequence of moves. After you type D, the prompts will ask you whether to use disk or tape to save or recall a cube. If you choose tape, you will be prompted to get the tape ready and then proceed. If you choose disk, you will be shown a catalog of the disk in the drive. You must then supply a file name for the cube to be saved or recalled. All file names for cubes end with ".CUBE", but you never need to type that (although you may if you wish). If you make any errors, you will be told, and either returned to the display or given the option of trying again. In no case can a typographical or disk error make you lose a cube in progress (whew!).

The Q command is used to quit, and is self-explanatory. If you should type it in error, it gives you a second chance to go back to where you were.

At the start of the program, you will have a choice of moving the face colors around and setting up each little cubelet. These facilities are there for you to use the program along with a real, solid cube. Just follow the self-explanatory prompts.

Variables

Note: In the variable names the letters F, R, and C usually stand for face, row, and column.

A\$: Input variable for yes/no answers and other inputs that are not needed later. Also used as a GET variable to put a pause between instruction blurbs. Also used in cover-page routine as a string of asterisks.

AC\$: Used to receive the user's abort/continue choice in the error-trapping routine.

AXIS: The number that corresponds to AXIS\$ (1 = x, 2 = y, 3 = z).

AXIS\$: This is the axis name (x, y, z) that the user inputs to the spin subroutine.

BEGIN%: First face to be drawn by the face-drawing subroutine.

C, C2, C7: Loop indices used to control the face number.

C\$: The portion of the command

currently being processed.

CC: Flags whether LP should

control the column number;

mnemonic for column change.

CC\$: This is extracted from the RULE\$ array, and is the variable that determines whether the upper, bottom, left, or right row or column of a face is to be moved or changed.

CLR%(n): Contains the color code numbers for each face. The subscript refers to the face.

CLR\$(n): Contains the names (e.g., orange, maroon) of the colors.

CM\$: The full command string as input by the user.

D\$: CHR\$(4), used for DOS commands.

DI\$: The directional symbol (+, -, 2) that the user inputs to the spin subroutine.

DI\$(i): Contains the directional symbols +, 2, and -.

E%: Error code gotten from PEEK(222).

F, F2, F7: Loop indices used to control the face number.

F\$(i): Contains the one-letter face names.

F3: Loop index used to reREAD CLR\$(F3) when each cubelet is set up manually.

FC: Loop index used to control the face during set-up of each cubelet.

FF, FT: Mnemonics for face-from and face-to. Derived from the RULE\$ and SRULE\$ arrays, they determine which face will have a row or column extracted into the TR% or RT% arrays or have a row or column replaced by the contents of TR% or RT%.

I: Used to seed the RND(1) function and as a general loop index.

KK: Loop index for pauses.

LP: Short for "loop." This loop index controls the value of R or C in the face-turning and cube-spinning subroutines. Whether R or C is controlled is determined by the flags RC and CC.

N: Number of turns required to accomplish a cube spin. Similar to NT.

N\$: File name for save/recall.

NT: Number of turns required to accomplish a face turn. 1, 2, and 3 correspond to +, 2, and -.

P%: Used to POKE into RAM the

Machine Language routine prescribed on page 136 of the Applesoft manual.

R, R2, R7: Loop indices used to control the row number in a face. R is also controlled by LP.

RC: Flags whether LP controls R. RESTART: Flags whether user has quit and requested a restart.

RESTART = 1 suppresses the on-screen instructions.

RT%(i): Used to hold a row or column from the cube in order to move it about.

RULE\$(i,j): Rule table for face rotations. The first subscript refers to the face.

Q: Loop index used to reduce the number of active FOR loops to an allowable quantity.

QUIT%: Last face to be drawn by the face-drawing subroutine.

SRULE\$(i,j): Rule table for whole-cube spins. The first subscript refers to the axis.

START%(i,j): Holds the starting x and y coordinates for each face on the Lo-Res screen. The first subscript refers to the face; the second, to x and y.

SX, SY: Starting x and y coordinates derived from the START% array.

TR%(i): Used to transfer a row or column from one part of the cube to another.

TRN: Loop index used to control the number of turns in a cube spin.

TURN: Loop index used to control the number of turns in a face turn. X: Loop variable used in drawing the gray background.

X%(i,j,k): This is the cube representation, the array that is manipulated and displayed on the screen. Its contents are color codes. The subscripts refer to the face, row, and column, respectively. For example, if the center cubelet of the F face is orange, then X%(5,2,2) = 9, because 9 is the code for orange in the Lo-Res mode.

XO: Loop index used to dissect CM\$.

XT%(i,j,k): Temporarily holds a copy of the X% array each time a string of two or more basic commands is entered. When a typographical error is encountered, the X% array is restored to its prior state with the XT% array.

```

#####
$  APPLSOFT BASIC      $
$  'RUBICUBE'         $
$  AUTHOR: F. J. CONDO JR. $
$  (C) 1982  SOFTSIDE  $
#####

```

Clears any mode that might garble or after the output, then blanks the screen.

```
15  NOTRACE : SPEED= 255: TEXT : HOME
```

Skips over the command-parsing subroutine to the beginning of the main program.

```
20  GOTO 250
```

COMMAND-PARSING SUBROUTINE

```
30  LC% = LEN (CM$): IF LC% = 0 GOTO
    150
```

Check for the special commands (S, D, Q).

```
40  C$ = LEFT$ (CM$,1)
50  IF C$ = "Q" GOTO 2650
60  IF C$ = "D" THEN GOSUB 2180:
    RETURN
70  IF C$ = "S" THEN GOSUB 1470:
    RETURN
```

Sets LC% equal to LEN(CM\$). If an odd number of characters were typed, LC% is decremented by one so that an even number of characters will be processed. This is because a basic command is always two characters.

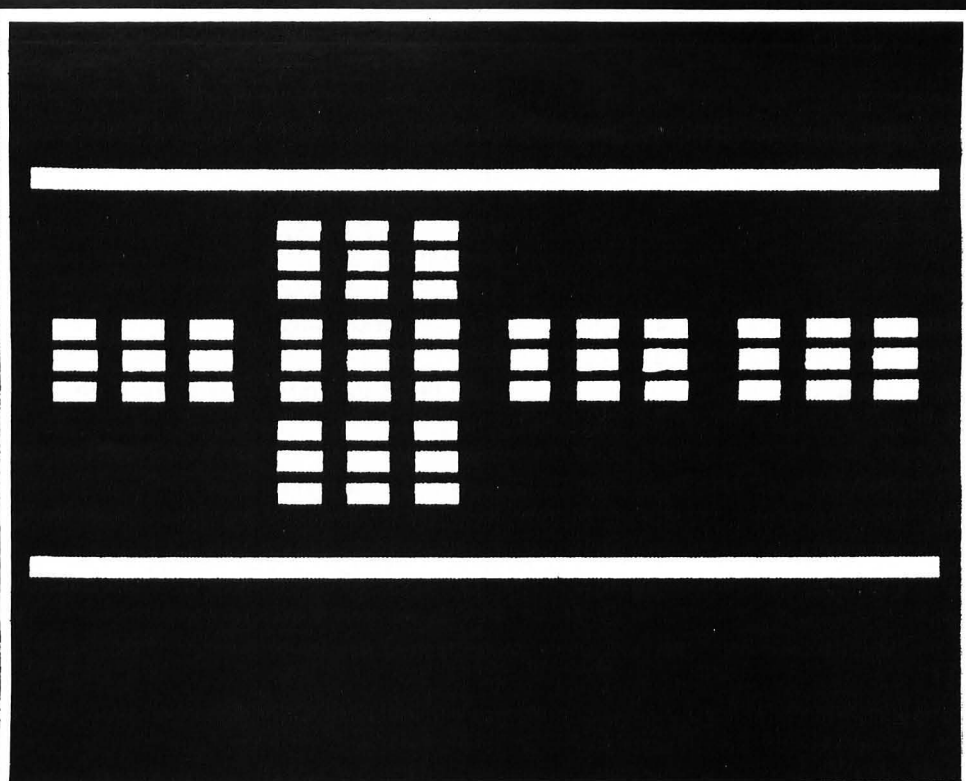
```
80  IF LC% / 2 < > INT (LC% / 2
    ) THEN LC% = LC% - 1
```

Branches to subroutine in line 230 when more than a single command was typed.

```
90  IF LC% > 2 THEN GOSUB 230
```

Examine CM\$ two characters at a time, and determine the face (F) and number of turns (NT) required, then branch to the face-turning subroutine at line 610. Since there are only six faces, F=7 flags an error. If an error occurs, array X% is restored from array XT%, and the program returns to the command prompt.

```
100 HTAB 16: FLASH : PRINT "WORK
```



```

ING";: NORMAL
110 FOR X0 = 1 TO LC% STEP 2
120 C$ = MID$ (CM$,X0,2)
130 FOR F = 1 TO 6: IF F$(F) = LEFT$
    (C$,1) GOTO 150
140 NEXT F
150 IF F = 7 THEN PRINT CHR$ (
    7);"ERROR";: FOR KK = 1 TO 1
    500: NEXT KK: PRINT
160 IF F = 7 AND LC% > 2 THEN GOSUB
    240
170 IF F = 7 THEN POP : GOTO 59
    0
180 FOR NT = 1 TO 3
190 IF RIGHT$ (C$,1) = DI$(NT) GOTO
    220
200 NEXT NT
210 F = 7: GOTO 150
220 GOSUB 610: NEXT X0: RETURN

```

Makes array XT% into a copy of array X%.

```
230 FOR F7 = 1 TO 6: FOR R7 = 1 TO
    3: FOR C7 = 1 TO 3: XT$(F7,R7
    ,C7) = X$(F7,R7,C7): NEXT C7
    ,R7,F7: RETURN
```

Makes array X% into a copy of array XT%.

```
240 FOR F7 = 1 TO 6: FOR R7 = 1 TO
    3: FOR C7 = 1 TO 3: X$(F7,R7,
    C7) = XT$(F7,R7,C7): NEXT C7
```

```
,R7,F7: RETURN
```

MAIN PROGRAM

Gets a seed for the random numbers from the locations randomized by the Apple Monitor's KEYIN subroutine.

```
250 I = PEEK (78) + 256 * PEEK
    (79): I = RND (- I)
```

Branches to the cover-page subroutine.

```
260 GOSUB 2710
```

Flags that the program is not being restarted from the QUIT command.

```
270 RESTART = 0
```

Sets D\$ to control-D for DOS commands.

```
280 D$ = CHR$ (4)
```

Dimension the arrays.

```
290 DIM X%(6,3,3),XT%(6,3,3),CLR
    %(6),CLR$(6),START$(6,2),F$(
    6),RULE$(6,8),SRULE$(3,10),R
    T%(3),TR%(3),DI$(3)
```

Color names are listed and read.

```
300 DATA "ORANGE", "BLUE", "MARDON
    ", "WHITE", "YELLOW", "GREEN"
310 FOR F = 1 TO 6: READ CLR$(F)
    : NEXT F
```

Rules table for face turns is listed and read.

```
320 DATA 2,5,4,6,U,U,U,1,6,3,5
    ,R,L,R,R,6,4,5,2,B,B,B,5,3
    ,6,1,L,L,R,L,1,2,3,4,B,L,U,R
    ,1,4,3,2,U,L,B,R
330 FOR F = 1 TO 6: FOR I = 1 TO
    8: READ RULE$(F,I): NEXT I,F
```

Face names are listed and read.

```
340 DATA "T","R","B","L","F","P"
350 FOR I = 1 TO 6: READ F$(I): NEXT
    I
```

Data for color codes of each face.

```
360 DATA 9,2,8,15,13,12
```

Data for starting locations for each face on the Lo-Res screen.

```
370 DATA 11,5,21,15,11,25,1,15,1
    1,15,31,15
```

Reads color codes.

```
380 FOR I = 1 TO 6: READ CLR$(I)
    : NEXT I
```

Reads starting locations.

```
390 FOR F = 1 TO 6: FOR I = 1 TO
    2: READ START$(F,I): NEXT I,
    F
```

Set each cubelet to the characteristic color for that face, that is, start out with a solved cube.

```
400 FOR F = 1 TO 6: FOR R = 1 TO
    3: FOR C = 1 TO 3
410 X$(F,R,C) = CLR$(F)
420 NEXT C,R,F
```

Directional symbols are listed and read.

```
430 DATA "+","2","-"
440 FOR I = 1 TO 3: READ DI$(I):
    NEXT I
```

Cube-spin rules table is listed and read.

```
450 DATA R+,L-,1,6,3,5,R,R,R,R,T
```

```
+ ,B-,2,5,4,6,C,C,C,C,F+,P-,1
,2,3,4,C,R,C,R
```

```
460 FOR AXIS = 1 TO 3: FOR I = 1
    TO 10: READ SRULE$(AXIS,I):
    NEXT I,AXIS
```

If this is the first time here, branch to the instructions.

```
470 IF NOT RESTART THEN GOSUB
    2770
```

Branches to face color changing subroutine.

```
480 GOSUB 2470
```

Branches to background-drawing subroutine.

```
490 GOSUB 2440
```

Sets up parameters to draw the whole cube, then branches to the cube-drawing subroutine.

```
500 BEGIN% = 1:QUIT% = 6: GOSUB 2
    390
```

Branches to the subroutine that pokes in Machine Language routine prescribed for use with ONERR GOTO on p. 136 of the Applesoft manual.

```
510 GOSUB 2960
```

Sets up a branch to the error-trapping routine in case of disk error.

```
520 ONERR GOTO 2970
```

Branches to the condensed-instructions subroutine.

```
530 GOSUB 2620
```

Sends user to the cubelet set-up subroutine if desired.

```
540 NORMAL
550 INPUT "DO YOU WANT TO SET UP
    EACH CUBELET?";A$:A$ = LEFT$(
    A$,1): IF A$ = "Y" THEN GOSUB
    3140: GOTO 590
```

Ask the user whether to scramble

the cube. Do 15 random face turns if scrambling is requested. This range of lines is skipped if each individual cubelet has been set up.

```
560 INPUT "DO YOU WANT THE CUBE
    SCRAMBLED?";A$:A$ = LEFT$(
    A$,1): IF A$ < > "Y" GOTO 5
    90
570 PRINT : FLASH : HTAB 15: PRINT
    "SCRAMBLING";: NORMAL
580 FOR I = 1 TO 15:CM$ = F$(INT
    (6 * RND (1) + 1)) + DI$(INT
    (3 * RND (1) + 1)):LC% = 2:
    GOSUB 110: NEXT I:BEGIN% =
    1:QUIT% = 6: GOSUB 2390
```

This is the main command prompt loop. It gets command strings, branches to the command-parsing subroutine, and draws the result on the screen.

```
590 PRINT
600 INPUT "COMMAND: ";CM$: GOSUB
    30:BEGIN% = 1:QUIT% = 6: GOSUB
    2390: GOTO 590
```

FACE-TURNING SUBROUTINE

This subroutine treats the turning of a face as two tasks: (1) move the cubelets of adjacent faces, and (2) rotate the face itself. The cubelets correspond to the elements of array X%.

NT sets the number of turns. A clockwise turn is defined by the RULE\$ table, so a counterclockwise turn is executed as three clockwise turns, and a 180-degree turn is executed as two clockwise turns.

```
610 FOR TURN = 1 TO NT
```

Step (1).
Move the first row or column into RT% for safekeeping.

```
620 FF = VAL (RULE$(F,1)):CC% =
    RULE$(F,5)
630 CC = 0:RC = 0
640 IF CC% = "U" THEN R = 1:CC =
    1: GOTO 680
650 IF CC% = "B" THEN R = 3:CC =
    1: GOTO 680
660 IF CC% = "L" THEN C = 1:RC =
    1: GOTO 680
```



```

670 C = 3:RC = 1
680 FOR LP = 1 TO 3
690 IF RC THEN R = LP
700 IF CC THEN C = LP
710 RT%(LP) = X%(FF,R,C)
720 NEXT LP
    
```

Move the second row or column into TR%, swap TR%(1) and TR%(3) if necessary (due to the behavior of real cubes), then move the second row or column into the place of the first row or column.

```

730 FF = VAL (RULE$(F,4)):CC$ =
    RULE$(F,8)
740 CC = 0:RC = 0
750 IF CC$ = "U" THEN R = 1:CC =
    1:GOTO 790
760 IF CC$ = "B" THEN R = 3:CC =
    1:GOTO 790
770 IF CC$ = "L" THEN C = 1:RC =
    1:GOTO 790
780 C = 3:RC = 1
790 FOR LP = 1 TO 3
800 IF RC THEN R = LP
810 IF CC THEN C = LP
820 TR%(LP) = X%(FF,R,C)
830 NEXT LP
840 FT = VAL (RULE$(F,1)):CC$ =
    RULE$(F,5)
850 CC = 0:RC = 0
860 IF CC$ = "U" THEN R = 1:CC =
    1:GOTO 900
870 IF CC$ = "B" THEN R = 3:CC =
    1:GOTO 900
880 IF CC$ = "L" THEN C = 1:RC =
    1:GOTO 900
890 C = 3:RC = 1
900 IF F = 5 THEN T% = TR%(1):TR
    %(1) = TR%(3):TR%(3) = T%
910 FOR LP = 1 TO 3
920 IF RC THEN R = LP
930 IF CC THEN C = LP
940 X%(FT,R,C) = TR%(LP)
    
```

```

950 NEXT LP
    
```

Beginning of IF loop that ends at line 1200. The structure of the RULE\$ table allows Q to control the next two steps via one sequence of BASIC coding.

```

960 Q = 0
    
```

Do the same for the third and fourth row or column as 730-950 do for the second row or column.

```

970 FF = VAL (RULE$(F,3 - Q)):CC$ =
    $ = RULE$(F,7 - Q)
980 CC = 0:RC = 0
990 IF CC$ = "U" THEN R = 1:CC =
    1:GOTO 1030
1000 IF CC$ = "B" THEN R = 3:CC =
    1:GOTO 1030
1010 IF CC$ = "L" THEN C = 1:RC =
    1:GOTO 1030
1020 C = 3:RC = 1
1030 FOR LP = 1 TO 3
1040 IF RC THEN R = LP
1050 IF CC THEN C = LP
1060 TR%(LP) = X%(FF,R,C)
1070 NEXT LP
1080 FT = VAL (RULE$(F,4 - Q)):C
    C$ = RULE$(F,8 - Q)
1090 CC = 0:RC = 0
1100 IF CC$ = "U" THEN R = 1:CC =
    1:GOTO 1140
1110 IF CC$ = "B" THEN R = 3:CC =
    1:GOTO 1140
1120 IF CC$ = "L" THEN C = 1:RC =
    1:GOTO 1140
1130 C = 3:RC = 1
1140 IF (Q = 0 AND F = 6) OR (Q =
    1 AND F = 5) OR (F = 4 AND Q
    = 1) OR (Q = 1 AND F = 2) OR
    (Q = 0 AND F = 4) THEN T% =
    TR%(1):TR%(1) = TR%(3):TR%(3
    ) = T%
    
```

```

1150 FOR LP = 1 TO 3
1160 IF RC THEN R = LP
1170 IF CC THEN C = LP
1180 X%(FT,R,C) = TR%(LP)
1190 NEXT LP
1200 IF Q = 0 THEN Q = 1:GOTO 9
    70
    
```

Put the first row or column, which was saved in RT%, into its new location.

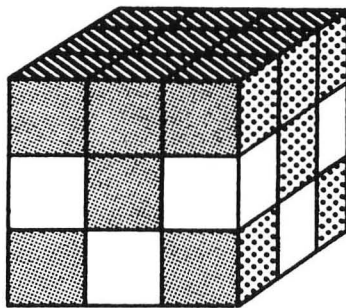
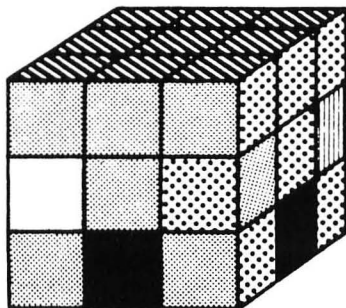
```

1210 FT = VAL (RULE$(F,2)):CC$ =
    RULE$(F,6)
1220 CC = 0:RC = 0
1230 IF CC$ = "U" THEN R = 1:CC =
    1:GOTO 1270
1240 IF CC$ = "B" THEN R = 3:CC =
    1:GOTO 1270
1250 IF CC$ = "L" THEN C = 1:RC =
    1:GOTO 1270
1260 C = 3:RC = 1
1270 IF F = 2 OR F = 6 THEN T% =
    RT%(1):RT%(1) = RT%(3):RT%(3
    ) = T%
1280 FOR LP = 1 TO 3
1290 IF RC THEN R = LP
1300 IF CC THEN C = LP
1310 X%(FT,R,C) = RT%(LP)
1320 NEXT LP
1330 NEXT TURN
    
```

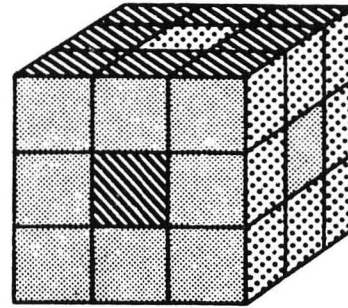
Step (2). This is accomplished in a manner exactly analogous with that of Step (1). The only difference is that the rules are the same regardless of the face, so there are no references to any rules table.

```

1340 FOR TURN = 1 TO NT
1350 R = 1:FOR C = 1 TO 3:RT%(C)
    = X%(F,R,C):NEXT C
1360 C = 1:FOR R = 1 TO 3:TR%(R)
    = X%(F,R,C):NEXT R
1370 T% = TR%(1):TR%(1) = TR%(3):
    
```



SoftSide



```

TRZ(3) = TZ
1380 R = 1: FOR C = 1 TO 3: X%(F,R
,C) = TRZ(C): NEXT C
1390 R = 3: FOR C = 1 TO 3: TRZ(C)
= X%(F,R,C): NEXT C
1400 C = 1: FOR R = 1 TO 3: X%(F,R
,C) = TRZ(R): NEXT R
1410 C = 3: FOR R = 1 TO 3: TRZ(R)
= X%(F,R,C): NEXT R
1420 TZ = TRZ(1): TRZ(1) = TRZ(3):
TRZ(3) = TZ
1430 R = 3: FOR C = 1 TO 3: X%(F,R
,C) = TRZ(C): NEXT C
1440 C = 3: FOR R = 1 TO 3: X%(F,R
,C) = RTZ(R): NEXT R
1450 NEXT TURN
1460 RETURN

```

CUBE-SPINNING SUBROUTINE

Determine the axis and direction, or display an error message.

```

1470 INPUT "SPIN ABOUT WHICH AXI
S (X,Y,Z)?:"; AXIS#: IF AXIS# =
"" GOTO 1520
1480 AXIS = ASC (AXIS#) - 87: IF
AXIS < 1 OR AXIS > 3 GOTO 15
20
1490 INPUT "WHICH DIRECTION (+,-
,2)?": DI#
1500 FOR N = 1 TO 3: IF DI# = DI
$(N) GOTO 1530
1510 NEXT N
1520 PRINT CHR$(7); "ERROR": RETURN

```

In a manner exactly analogous with that of the face-turning subroutine, the cubelets of the central row or column between the two faces are moved.

```

1530 PRINT : FLASH : HTAB 10: PRINT
"SPINNING ENTIRE CUBE": NORMAL
1540 TRN = 0
1550 TRN = TRN + 1: IF TRN > N GOTO
2170
1560 FOR K = 1 TO 2: CM# = SRULE#
(AXIS,K): LC# = 2: GOSUB 110:
NEXT K
1570 FF = VAL (SRULE$(AXIS,3)): C
C# = SRULE$(AXIS,7)
1580 CC = 0: RC = 0
1590 IF CC# = "C" THEN CC = 1: R =
2: GOTO 1610
1600 RC = 1: C = 2
1610 FOR LP = 1 TO 3

```

```

1620 IF CC THEN C = LP
1630 IF RC THEN R = LP
1640 RTZ(LP) = X%(FF,R,C)
1650 NEXT LP
1660 FF = VAL (SRULE$(AXIS,6)): C
C# = SRULE$(AXIS,10)
1670 CC = 0: RC = 0
1680 IF CC# = "C" THEN CC = 1: R =
2: GOTO 1700
1690 RC = 1: C = 2
1700 FOR LP = 1 TO 3
1710 IF CC THEN C = LP
1720 IF RC THEN R = LP
1730 TRZ(LP) = X%(FF,R,C)
1740 NEXT LP
1750 IF AXIS = 3 THEN TZ = TRZ(3
): TRZ(3) = TRZ(1): TRZ(1) = T
Z
1760 FT = VAL (SRULE$(AXIS,3)): C
C# = SRULE$(AXIS,7)
1770 CC = 0: RC = 0
1780 IF CC# = "C" THEN CC = 1: R =
2: GOTO 1800
1790 RC = 1: C = 2
1800 FOR LP = 1 TO 3
1810 IF CC THEN C = LP
1820 IF RC THEN R = LP
1830 X%(FT,R,C) = TRZ(LP)
1840 NEXT LP
1850 Q = 0
1860 FF = VAL (SRULE$(AXIS,5 - Q
)): CC# = SRULE$(AXIS,9 - Q)
1870 CC = 0: RC = 0
1880 IF CC# = "C" THEN CC = 1: R =
2: GOTO 1900
1890 RC = 1: C = 2
1900 FOR LP = 1 TO 3
1910 IF CC THEN C = LP
1920 IF RC THEN R = LP
1930 TRZ(LP) = X%(FF,R,C)
1940 NEXT LP
1950 IF Q = 1 AND (AXIS = 1 OR A
XIS = 3) THEN TZ = TRZ(1): TR
Z(1) = TRZ(3): TRZ(3) = TZ
1960 FT = VAL (SRULE$(AXIS,6 - Q
)): CC# = SRULE$(AXIS,10 - Q)
1970 CC = 0: RC = 0
1980 IF CC# = "C" THEN CC = 1: R =
2: GOTO 2000
1990 RC = 1: C = 2
2000 FOR LP = 1 TO 3
2010 IF CC THEN C = LP
2020 IF RC THEN R = LP
2030 X%(FT,R,C) = TRZ(LP)
2040 NEXT LP

```

```

2050 IF Q = 0 THEN Q = 1: GOTO 1
860
2060 IF AXIS = 1 THEN TZ = RTZ(1
): RTZ(1) = RTZ(3): RTZ(3) = T
Z
2070 FT = VAL (SRULE$(AXIS,4)): C
C# = SRULE$(AXIS,8)
2080 CC = 0: RC = 0
2090 IF CC# = "C" THEN CC = 1: R =
2: GOTO 2110
2100 RC = 1: C = 2
2110 FOR LP = 1 TO 3
2120 IF CC THEN C = LP
2130 IF RC THEN R = LP
2140 X%(FT,R,C) = RTZ(LP)
2150 NEXT LP
2160 GOTO 1550
2170 PRINT : RETURN

```

SAVE/RECALL SUBROUTINE. The array X% is the entity saved or recalled.

```

2180 INPUT "DO YOU WANT TO SAVE
OR RECALL?": C#: C# = LEFT$ (
C#,1)
2182 IF C# < > "S" AND C# < >
"R" THEN PRINT CHR$(7); "E
RROR": FOR KK = 1 TO 1800: NEXT
KK: RETURN
2184 INPUT "USING DISK OR TAPE?"
; CC#: CC# = LEFT$ (CC#,1)
2186 IF CC# = "D" THEN 2210
2188 IF CC# < > "T" THEN 2184
2190 TEXT : HOME : IF C# = "R" THEN
2200
2192 INPUT "POSITION TAPE, START
IN 'RECORD' MODE, AND THEN
PRESS 'RETURN' ": C#
2194 STORE X%
2196 GOTO 2290
2200 INPUT "POSITION TAPE, START
IN 'PLAY' MODE, AND PRES
S 'RETURN' ": C#
2202 RECALL X%
2204 GOTO 2290
2210 TEXT : HOME : IF C# = "R" GOTO
2300
2220 PRINT D#: "CATALOG"
2230 INPUT "FILE NAME FOR SAVE:
": N#
2240 IF RIGHT$ (N#,5) = ".CUBE"
THEN N# = LEFT$ (N#, LEN (
N#) - 5)
2250 PRINT D#: "OPEN": N#: ".CUBE":

```

```

PRINT D$;"WRITE";N$;".CUBE"
2260 FOR F = 1 TO 6: FOR R = 1 TO
3: FOR C = 1 TO 3
2270 PRINT X%(F,R,C): NEXT C,R,F
2280 PRINT D$;"CLOSE"
2290 GOSUB 2440: GOSUB 2620: RETURN
2300 PRINT D$;"CATALOG"
2310 INPUT "FILE NAME TO RECALL?"
:N$
2320 IF RIGHT$(N$,5) = ".CUBE"
THEN N$ = LEFT$(N$, LEN (
N$) - 5)
2330 PRINT D$;"VERIFY";N$;".CUBE"
"
2340 PRINT D$;"OPEN";N$;".CUBE":
PRINT D$;"READ";N$;".CUBE"
2350 FOR F = 1 TO 6: FOR R = 1 TO
3: FOR C = 1 TO 3
2360 INPUT X%(F,R,C): NEXT C,R,F
2370 PRINT D$;"CLOSE"
2380 GOSUB 2440: GOSUB 2620: RETURN

```

FACE-DRAWING SUBROUTINE.

```

2390 FOR F = BEGIN% TO QUIT%:SX =
START%(F,1):SY = START%(F,2)
2400 FOR R = 1 TO 3: FOR C = 1 TO
3: COLOR= X%(F,R,C)
2410 X = SX + 3 * (C - 1):Y = SY +
3 * (R - 1): FOR KK = 0 TO 1
: HLIN X,X + 1 AT Y + KK: NEXT
KK
2420 NEXT C,R,F
2430 RETURN

```

BACKGROUND-DRAWING SUBROUTINE.

```

2440 GR : COLOR= 14
2450 FOR I = 0 TO 2: HLIN 0,39 AT
I: HLIN 0,39 AT 39 - I: NEXT
2460 COLOR= 5: FOR X = 2 TO 37: HLIN
0,39 AT X: NEXT X: RETURN

```

ALTER THE FACE COLORS to match a real cube.

```

2470 TEXT : HOME : VTAB 10
2480 PRINT "NOT ALL CUBES ARE CO
LORED THE SAME WAY. YOURS MA
Y BE DIFFERENT FROM THE COMP
UTERMODEL. IF YOU WANT TO US
E THE PROGRAM INCONJUNCTION
WITH A REAL CUBE, YOU WILL
WANT TO ASSIGN THE COLORS PR
OPERLY."
2490 PRINT : PRINT : INPUT "DO Y
OU WANT TO ALTER THE FACE CO
LORS?";A$:A$ = LEFT$(A$,1)
: IF A$ < > "Y" THEN HOME
: RETURN
2500 HOME : PRINT "SET YOUR CUBE
BEFORE YOU. I WILL NOW ASKY
OU THE NAME OF THE COLOR ON
THE CENTRALCUBELET OF EACH O
F THE SIX FACES. HEREARE T
HE COLORS I KNOW:"

```

Print the color names.

```

2510 PRINT : FOR F = 1 TO 6: PRINT
TAB( 16);CLR$(F): NEXT F

```

Preserve the top 13 lines of the screen, so that the color names stay on the screen.

```

2520 POKE 34,13: HOME

```

Ask for the color for each face. If the input is not recognized, ask again. After a color has been recognized, change its name to the code of the ESC key (CHR\$(27)), so that it won't be recognized again. If a used color is typed, ask again.

```

2530 FOR F = 1 TO 6
2540 PRINT "WHAT COLOR FOR THE "

```

```

;F$(F);" FACE";
2550 INPUT A$
2560 FOR FF = 1 TO 6
2570 IF LEFT$(A$,1) = LEFT$(
CLR$(FF),1) THEN CLR$(FF) =
CHR$(27): GOTO 2590
2580 NEXT FF: GOTO 2540

```

Set each face in turn to the selected color.

```

2590 FOR R = 1 TO 3: FOR C = 1 TO
3
2600 X%(F,R,C) = CLR$(FF)
2610 NEXT C,R,F: TEXT : HOME : RETURN

```

PRINT CONDENSED INSTRUCTIONS at the bottom of the screen in inverse video. The POKE allows only the bottom-most line on the screen to change.

```

2620 INVERSE
2630 HOME : PRINT "+:CLOCKWISE
(T) -:COUNTERCLOCKWISE2
:180 DEGREES (L)(F)(R)(P)
Q:QUITS:SPIN CUBE
(B) D:SAVE/RECALL";
2640 NORMAL : POKE 34,23: RETURN

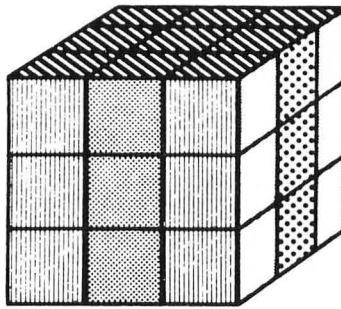
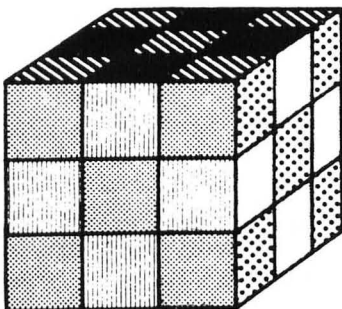
```

QUIT COMMAND. Allows user to cancel the quit, to restart, or really quit. The POP allows the GOTO 310 to replace the normal RETURN.

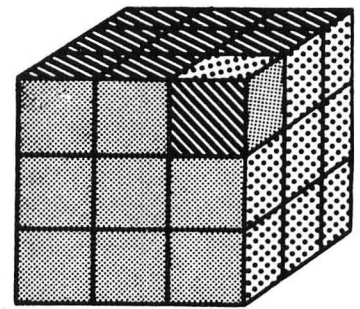
```

2650 TEXT : HOME : VTAB 12: INPUT
"DO YOU WANT TO QUIT, RESTAR
T, OR GO BACKTO WHERE YOU WE
RE (Q, R, OR G)? ";A$:A$ = LEFT$(
A$,1)
2660 IF A$ = "Q" THEN HOME : END
2670 IF A$ = "G" THEN GOSUB 244
0: GOSUB 2620: RETURN
2680 IF A$ < > "R" GOTO 2650
2690 RESTART = 1

```



SoftSide



```
2700 RESTORE : POP : GOTO 310
```

Display the title page and copyright notice.

```
2710 A$ = "*****
*****"
2720 VTAB 10: PRINT A$: VTAB 14:
PRINT A$
2730 VTAB 12: HTAB 13: PRINT "R
U B I C U B E"
2740 FOR KK = 1 TO 500: NEXT KK
2750 VTAB 16: HTAB 5: PRINT "COP
YRIGHT F. J. CONDO JR. 1981"
2760 RETURN
```

Display the disclaimer and then display the introductory instructions, if desired. This routine does not come up when restart has been chosen.

```
2770 VTAB 20: PRINT "RUBIK'S CUB
E IS A TRADEMARK OF IDEAL TO
YCORPORATION, WHICH IS NO
T ASSOCIATED WITH
THIS PROGRAM"
2780 FOR KK = 1 TO 3000: NEXT KK
2790 HOME : VTAB 10: INPUT "DO Y
OU WANT INSTRUCTIONS?";A$:A$
= LEFT$(A$,1): IF A$ < >
"Y" THEN RETURN
2800 HOME : VTAB 10: PRINT "THE
RUBIK'S CUBE CRAZE IS SWEEP
ING THE NATION. THIS PROGRAM
SIMULATES THE CUBE AND ITS
TWISTS AND TURNS IN THE APPL
E'S LOW RESOLUTION GRAPHICS.
TO SEE WHAT THECUBE LOOKS L
IKE, HIT A KEY";: GET A$
2810 GOSUB 2440:BEGIN% = 1:QUIT%
= 6: GOSUB 2390
2820 HOME : PRINT "A CUBE HAS SI
X FACES, EACH WITH A UNIQUEC
HARACTERISTIC COLOR. THE CE
NTER SQUAREDETERMINES THE C
HARACTERISTIC COLOR OFEACH
FACE. HIT A KEY TO GO ON.";
: GET A$
2840 HOME : PRINT "THE SYMBOL FO
R CLOCKWISE IS + T
HE SYMBOL FOR COUNTERCLOCKWI
SE IS - THE SYMBOL FOR 18
0 DEGREES ABOUT IS 2 DON'T
WORRY. I'LL REMIND YOU. HIT
A KEY";: GET A$
```

```
2860 HOME : PRINT "EACH FACE HAS
A ONE LETTER NAME: F)FRONT,B
)ACK, T)OP, B)OTTOM, L)EFT,
R)IGHT, ANDP)OSTERIOR. HIT
ANY KEY TO CONTINUE ";: GET
A$
2930 HOME : PRINT "FOR FULL INS
TRUCTIONS, SEE THE ARTICLEI
N S O F T S I D E M A G A
Z I N E !!": FOR KK = 1 TO
2500: NEXT KK
2940 RETURN
```

The Machine Language routine prescribed on p. 136 of the Applesoft manual is POKEd into memory.

```
2950 DATA 104,168,104,166,223,15
4,72,152,72,96
2960 FOR I = 768 TO 777: READ P%
: POKE I,P%: NEXT I: RETURN
```

ERROR-TRAPPING ROUTINE.

Line 2970: CALLs the Machine Language routine from p. 136 of the Applesoft manual.

```
2970 CALL 768
```

E% is set to the error code. Both Applesoft and DOS store this code in PEEK(222). For the meanings of the error codes, see p. 136 of the Applesoft manual and p. 114 of the DOS manual.

```
2980 E% = PEEK (222)
```

Since control-C is considered an error by Applesoft, this line is needed to allow a control-C ending of execution.

```
2990 IF E% = 255 THEN TEXT : HOME
: PRINT "CTRL-C IN LINE "; PEEK
(218) + 256 * PEEK (219); CHR$
(7): END
```

Print message appropriate for the error. All of these are disk errors.

```
3000 PRINT CHR$ (7);"FILE ";N$;
".CUBE ";
3010 IF E% = 5 THEN PRINT "IS D
EFFECTIVE.": GOTO 3100
3020 IF E% = 11 THEN PRINT : PRINT
```

```
CHR$ (7);"DOS SYNTAX ERROR.
": GOTO 3100
3030 IF E% = 6 THEN PRINT "NOT
FOUND.": GOTO 3100
3040 IF E% = 13 THEN PRINT "IS
NOT A TEXT FILE.": GOTO 3100
3050 IF E% = 8 THEN PRINT : PRINT
"I/O ERROR"; CHR$ (7): GOTO
3100
3060 IF E% = 9 THEN PRINT "WON'
T FIT ON THIS DISK.": GOTO 3
100
3070 IF E% = 10 THEN PRINT "IS
LOCKED.": GOTO 3100
3080 IF E% = 4 THEN PRINT : PRINT
"CAN'T BE SAVED ON WRITE-PRD
TECTED DISK.": GOTO 3100
```

This line will never be executed, but if you are manually typing the program, it will allow you to detect any typographical errors. The PEEKs are where Applesoft stores the line number of the line that contained an error.

```
3090 TEXT : HOME : VTAB 12: PRINT
CHR$ (7);"FATAL ERROR #";E%
;" IN LINE "; PEEK (218) + PEEK
(219) * 256: END
```

Determine whether to abort or continue (try again). Then branch to the appropriate part of the SAVE/RECALL subroutine.

```
3100 INPUT "TYPE ABORT OR CONTIN
UE: ";AC$
3110 AC$ = LEFT$(AC$,1)
3120 IF AC$ = "A" THEN GOTO 238
0
3130 GOTO 2184
```

SUBROUTINE TO SET UP EACH INDIVIDUAL CUBELET.

ReREAD color names, which were destroyed if face colors were altered.

```
3140 RESTORE : FOR F3 = 1 TO 6: READ
CLR$(F3): NEXT
```

Set each cubelet to a neutral color, except for the central cubelet on each face, which is its characteristic color.

```

3150 FOR F2 = 1 TO 6: FOR R2 = 1
      TO 3: FOR C2 = 1 TO 3: IF R
      2 = 2 AND C2 = 2 GOTO 3170
3160 XX(F2,R2,C2) = 14
3170 NEXT C2,R2,F2
  
```

Redraw the background, list the color names at the bottom of the screen, and display all but the first face with the neutral cubelets.

```

3180 GOSUB 2440: GOSUB 3300: BEGI
      N% = 2: QUIT% = 6: GOSUB 2390
  
```

For each face, set each cubelet in turn (except the central one) to black, and ask for the actual color. If the color name is not recognized, ask again, else redraw the face with the changed cubelet, and make the next cubelet black.

```

3190 FOR F2 = 1 TO 6: FOR R2 = 1
      TO 3: FOR C2 = 1 TO 3
3200 IF R2 = 2 AND C2 = 2 GOTO 3
      270
3210 XX(F2,R2,C2) = 0
3220 BEGIN% = F2: QUIT% = F2: GOSUB
      2390
3230 INPUT "COLOR NAME FOR THE D
      ARK CUBELET: "; A$
3240 FOR FC = 1 TO 6: IF LEFT$
      (A$,1) = LEFT$(CLR$(FC),1)
      THEN 3260
3250 NEXT FC: GOTO 3230
3260 XX(F2,R2,C2) = CLR$(FC)
  
```

In this line, BEGIN% and QUIT% are always equal, so that only the face in question is redrawn. This saves time.

```

3270 NEXT C2,R2: BEGIN% = F2: QUIT
      % = F2: GOSUB 2390: NEXT F2:
  
```

```

BEGIN% = 6: QUIT% = 6: GOSUB
      2390
  
```

If the user catches a mistake, start over.

```

3280 INPUT "IS THIS CORRECT?"; A$
      : A$ = LEFT$(A$,1): IF A$ =
      "N" GOTO 3150
  
```

Allow all four of the bottom text lines to be changed.

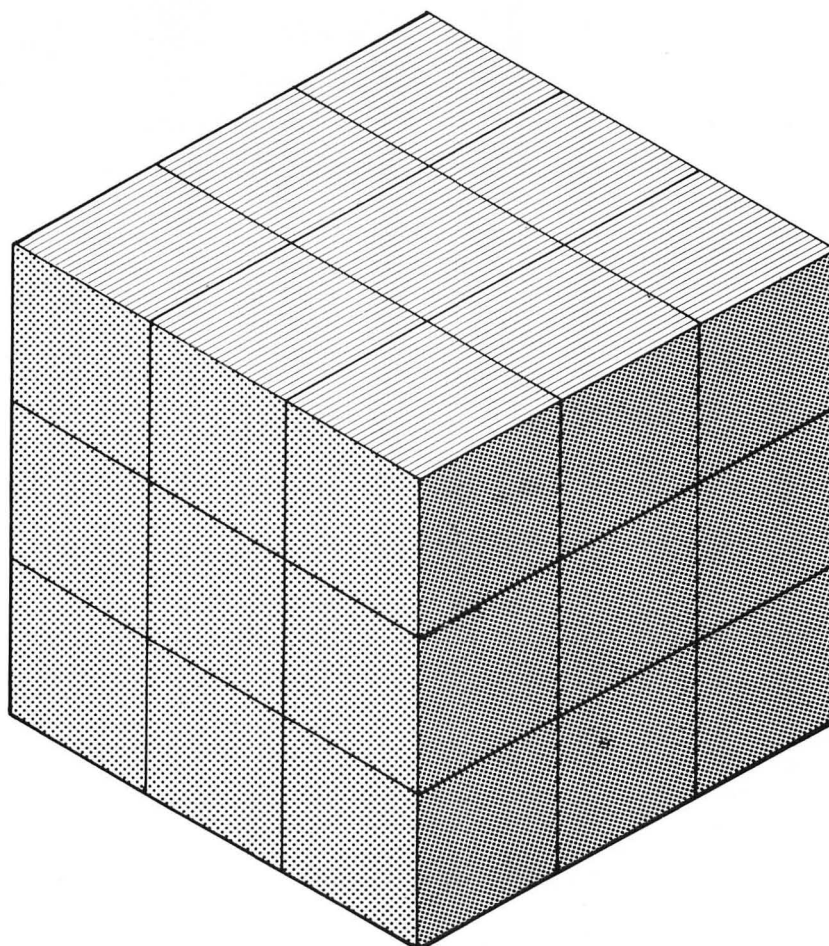
```

3290 POKE 34,20: GOSUB 2620: RETURN
  
```

Print the color names for reference. The POKE freezes the names on the screen, with the bottom-most line free to change.

```

3300 FOR F3 = 1 TO 3: PRINT CLR$(
      F3), CLR$(3 + F3): NEXT F3: POKE
      34,23: RETURN
  
```



Hi-Res Secrets

A review by Cary W. Bradley

By Don Fudge (Avant-Garde Creations) System requirements: 48K Apple with Applesoft. Suggested retail price: four disks and manual — \$125.

There's no question that the Apple II's high-resolution graphics capability is one of its niftiest features. It probably was among the reasons you chose to buy an Apple in the first place; I know it was for me. Just three letters — HGR — strategically located in a BASIC program or entered directly through the keyboard, transport you to that magical world of Hi-Res, where all manner of lines, patterns and shapes appear, disappear, reappear, and move, right before your very eyes.

Simple, right?

Well, not exactly. It probably didn't take you long to discover that the Apple manuals cover only the bare essentials of Hi-Res graphics. Sure, it's easy enough to go into HGR, choose HCOLOR, plot points, draw lines, and generate pretty patterns. After all, the programs to do them are all right there in the manual. But what about the trickier stuff, like shapes? Go ahead, try it the way the manual tells you. First, plot some shapes on graph paper. Now translate them into series of little arrows, express the arrows in binary code, group the 1s and 0s into appropriate bytes, retranslate the binary bytes into hexadecimal ones, count all of the bytes in every shape, set up a shape table, enter it all into memory, and save it on tape or disk. (Whew!)

If you go through all of this for anything but the simplest of shapes, you'll probably begin to seriously question whether you really need Hi-Res graphics in your programs. Of course, you can simplify the process quite a bit by using any of a number of available programs which automatically produce shape tables for you. Still, you've got to



write a program with lots of DRAWing and XDRAWing to make your shapes do anything. Compare the results of your hours of work to the graphics in any inexpensive arcade-type game, and you'll quickly realize that there's a very large gap between what you know about Apple graphics and what the professional programmers know.

Don't despair. *Hi-Res Secrets* was written for the sole purpose of bridging that gap. This extensive instructional package could just as appropriately have been called *Everything You Always Wanted to Know About High-Resolution Graphics, But Didn't Have Any Idea Who To Ask*. The system consists of four disks (over 200 files) and a 263-page manual. There's little doubt that it's the most complete compilation of Apple graphics information on the market. If you're

serious about using your Apple's Hi-Res capabilities, it's well worth the price. *Hi-Res Secrets* lets you in on advanced techniques that the experts have been using to make your Apple do those astounding Hi-Res feats. You could spend a long time playing around with what your Apple manuals tell you about Hi-Res graphics, and not even come close to discovering the tools and tricks you'll find in this comprehensive package.

The reason the author gives for having gone to the trouble of putting this all together is hard to argue with: considering the rapidly growing competition in the microcomputer market, the Apple is likely to be more viable, and for a longer time, if advanced programming techniques are shared widely, rather than being guarded by those relative few currently capable of producing high-quality Apple graphics soft-

ware. *Hi-Res Secrets* eliminates the need for the aspiring Apple programmer to rediscover, from scratch, what the pros already know about Hi-Res graphics.

There's no shortage of software that lets you create and manipulate Hi-Res shapes and pictures, but a quick perusal of the *Hi-Res Secrets* manual reveals a big difference. This one not only lets you do these things, it also explains how they're done, in detail, so that you can work with Hi-Res on your own terms. Many of the programs are listed and documented in the manual, including assembler source codes for Machine Language routines. Everything in the package is totally accessible! Two of the four disks are not copy-protected, so you can easily transfer their contents to any programming environment. What's more, programs which allow you to list and examine source files on the protected disks are included. Much of the material is designed to be incorporated directly into your own programs, and Avante-Garde Creations requests only that you acknowledge the fact that the routines came from their package, a fair deal if you plan to market your programs.

Hi-Res Secrets is for the serious programmer, or the person who wishes to become a serious programmer. If your software requirements are limited to programs which you simply boot up, respond to a prompt, sit back and watch the show, then read no further. This one requires work — a lot of it. But if you ever dreamed that you might be the next one to turn the graphics software market on its ear, you'd better get used to the fact that a lot of hard work will be involved. Since you're still reading, I'll assume that you realize this fact. With *Hi-Res Secrets*, Don Fudge has already done the first part of your work for you, namely the considerable task of discovering how to make your Apple perform the graphics feats it's capable of doing. You can spend less time inventing graphics techniques and more time creating that dynamite program that's trapped inside your head.

The decision as to whether *Hi-Res Secrets* is for you should be based

on two considerations: first, whether the kinds of graphics techniques presented are needed to accomplish your programming objectives, and second, whether you've got the time and desire to master them. If you just want to draw pretty pictures, there are cheaper packages that will do the job for you. Keep in mind, however, that with that type of software you will be limited to using your graphics in only the ways the author has defined. If you can live with that, fine. But with *Hi-Res Secrets*, these limitations are removed. Understanding is the goal, rather than convenience.

The package is presented in tutorial fashion, guiding you through the basic principles of shape-building and various types of animation. First, you're introduced to three different kinds of Apple shapes that form the basis for most of the graphics techniques in the system: vector shapes, hplot shapes and block shapes. (A quick word of explanation for those who are unfamiliar with the terms: Vector shapes are the ones with the little arrows and shape tables you'll find described in the Apple manuals. Hplot shapes consist of straight lines, for which you define the endpoints and call Apple's built-in HPLLOT routine. Block shapes are formed by defining bits to be turned "on" or "off" within the Hi-Res screen area of the Apple's memory.) The advantages and disadvantages of each are discussed, as well as the types of situations in which you'll be using each of them. This all becomes increasingly clear to you as you work through the examples, creating your own shapes and working with the samples provided for you.

Programs are included which allow you to define block shapes and hplot shapes. There is no utility for drawing vector shapes; software to do this is already available, from Avante-Garde Creations and others. Because some shapes are more easily defined in one form and better utilized in another, there are also programs which convert one type of shape to another, for all possible combinations of shape types you could reasonably expect to use. You can also load in shapes you've

created through other sources and apply any appropriate techniques to them.

After you've mastered the various types of shapes, you'll learn how to animate them, with a variety of animation techniques that covers just about any imaginable application. Step by step, you're guided through simple animation (just moving a shape from one location to another) and the more sophisticated technique of flipping back and forth between the two Hi-Res pages. Again, the examples are profuse, showing you cases of both good and bad animation sequences. You'll be able to animate your shapes using Applesoft, Machine Language, or a combination of the two.

The shape creation/animation group of programs constitutes a thorough self-study course in this area of Apple programming, and may well be worth the total package price by itself. *Hi-Res Secrets* doesn't stop there, though. The disks are loaded with a number of other utilities to further your education in the use of both Hi-Res graphics and sound routines. And the manual, in addition to documenting the disks' contents, includes discussions of miscellaneous programming topics that can be of great help when you finally begin to utilize your newly acquired skills. There are specific tips on using the routines in your own programs, controlling speed and timing, game paddle and keyboard inputs, making Integer Machine Language programs work properly from Applesoft, memory organization, Assembly Language programming, use of the "collision counter," and an extensive listing of PEEKs, POKEs and CALLs to help you take advantage of what's already packed inside your Apple.

A very good color-filling utility is included in the *Hi-Res Secrets* system. You can get this capability in other packages, but it's doubtful that any of them give you the flexibility of use that *Hi-Res Secrets* does. The color-filling routines are the fastest I've seen. This doesn't make a tremendous difference in the amount of time it takes to color-fill a picture, but is an indication of the care that went into the program-

ming of this package. The Machine Language routines are thoroughly documented, along with explanations of how color-filling is done. You can include a lot more graphics in your programs if you draw and color-fill them under program control, rather than just loading a 34-sector binary file every time you want to fill the Hi-Res screen. Extra features associated with this utility allow you to filter colors and fatten up thin white lines to give your pictures a more finished appearance.

You also get *Superfont*, a graphics utility which creates title pages, etc., with great big letters of ten different styles on the Hi-Res screen. This program is far superior to a similar one in my library. The quality is especially noticeable when you enlarge the letters. Of particular interest is the documentation and discussion of the techniques for scrolling these oversized letters, something that is done, but not explained, in other packages.

The sound portion of the system isn't nearly as complete as the graphics part (you can't have everything) but is worth mentioning. Most useful are nine Machine Language files that contain sounds that will add a nice touch to games. A program to let you write/record/play music is included, but could just as well have been left out. Play the sample tune that comes with it (not bad, Don) and if you have any sense of rhythm at all you'll notice right away that there's an annoying pause between screenfuls of notes. It's all right as a demonstration of sound routine use, but its practical value is marginal. An added bonus is a routine that creates "violin" sounds, rather than the plain tones you get from the simpler sound routines. It's not a part of the music-playing program, but you could probably incorporate it with a little effort. More useful is the capability to add it to your own programs, which you can do.

Graphics and sound are combined in a sample game, *The Minefields of Normalcy*, which is another demonstration of the techniques presented in the package, and mildly entertaining. You can ignore most of the instructions; they're about

90% longer than you need to be able to play.

In all, it's the graphics utilities and documentation that make *Hi-Res Secrets* worthwhile; the rest is just decoration. Viewed from that perspective, it represents a significant contribution to the growing class of software and literature aimed at helping you gain deeper insights into the inner workings of your Apple.

The matter of Machine Language is worth some further comments, especially if you're a beginner. You're probably aware that most of the best-quality graphics software consists mainly, or completely, of Machine Language. You'll almost certainly need to tackle this problem, with an assembler, if you plan to produce this kind of software. If you already know Assembly Language, you'll get a lot more out of *Hi-Res Secrets*. If you don't, you'll need to do some additional work in this area, and *Hi-Res Secrets* will provide a useful supplement to your study.

The assembler source files in *Hi-Res Secrets* were produced with the *LISA* assembler. You don't need an assembler to use the routines in this package, because the Machine Language versions are already there. But if you want to modify any of the routines for your application, you will need an assembler (and the ability to use it). If you don't have *LISA*, all is not lost. I used the *Apple Editor/Assembler* (from the DOS Tool Kit) on some of the *LISA*-generated files as a test, and found that with a minimum of editing, the files assembled with no problem. Chances are, whatever assembler you use, you'll be able to take advantage of this unique feature of *Hi-Res Secrets*. The assembler source files documented in the manual are described line by line, which you'll appreciate if you go in this direction.

Although the *Hi-Res Secrets* manual is extremely thorough, there are a few things about it that may annoy you a little after you've used it a while. One must appreciate the problem of putting together such a broad document for an audience so varied in expertise. The tutorial format does have some advantages over a dry reference manual. A

lighthearted approach certainly makes for easier reading as you go step by step through new territory. But after you've read it once or twice, and just want to use the techniques, the lines about Mexican pigeons roosting on your just-waxed T-Bird and the challenges to figure out what "ret up moc elppa" means (really, now) only serve to get in the way. It would be nice if there were a true reference manual to fall back on, rather than just the crowded pages of narrative that helped you in the initial learning process. Perhaps this will be added in the future.

The manual is intended to serve everyone from the rank beginner to the experienced programmer, which admittedly made it a difficult writing task. The approach taken to this problem is to attempt to reach every level of reader in every discussion. Particularly in the introductory material on shapes, this makes it difficult to get a foothold. In its present form, you'll catch on after several rereadings, but it would be more effective if the elementary stuff were separated from the more advanced material. This would allow the reader to concentrate on the appropriate level of information, and to skim over that which is above or below him.

None of this should obscure the fact that the *Hi-Res Secrets* manual is absolutely packed with valuable information. Any inconveniences in using the manual are vastly overshadowed by the tremendous volume of knowledge you can acquire through conscientious study. You can continue learning for months, limited only by your own ambition. The techniques and programs will never lose their usefulness.

The combination of instructional material and utilities to make use of the techniques puts this package in a class by itself. *Hi-Res Secrets* is bound to increase the ranks of good Apple graphics programmers by a substantial margin. Unless you're one of the select few who already qualify, and afraid of a little competition, this is good news. Whatever your present level of expertise, if your programming objectives include the use of high-quality graphics there's no better way to start than with *Hi-Res Secrets*. 5

K-Byter

System Configuration Test

An ATARI® K-Byter by Alan J. Zett, *SoftSide*

System Configuration Test is a helpful and simple utility program which will tell you at a glance what peripherals are attached to your ATARI® AND responding properly. Its use is simple: Just RUN it and it will give you a full report. If you have disk drives connected, you should insert a disk (anything with at least one program on it) in each drive before RUNNING the test. Because of the difficulty of testing the cassette recorder, it is assumed to be connected and working.

This would be an excellent utility to incorporate in a boot-up program, to let you know if everything is working properly whenever you power up.

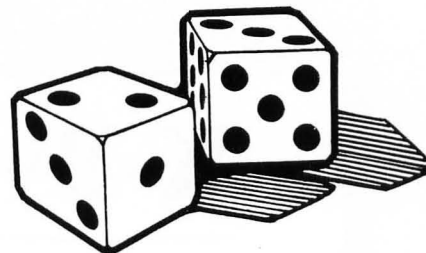
```
10 POKE 82,5:GRAPHICS 0:POKE 752,1:DIM
W$(29):W$="_____
```

```
___":? W$:N0=0:N3=3:N6=6:N12=12
20 ? "_ SYSTEM CONFIGURATION TEST_":?
W$
30 ? "_ By Alan J. Zett _":?
W$
40 POKE 82,11:?? W$(13):? "_ WORKING
UNITS_":? W$(13)
50 ? "_CASSETTE _":TRAP 60:CLOSE
#N3:OPEN #N3,N6,NO,"D1:$.*":? "_DISK
DRIVE #1_"
60 TRAP 70:CLOSE #N3:OPEN #N3,N6,NO,"D
2:$.*":? "_ DISK DRIVE #2_"
70 TRAP 80:CLOSE #N3:OPEN #N3,N6,NO,"D
3:$.*":? "_ DISK DRIVE #3_"
80 TRAP 90:CLOSE #N3:OPEN #N3,N6,NO,"D
4:$.*":? "_ DISK DRIVE #4_"
90 TRAP 100:CLOSE #3:OPEN #N3,8,NO,"P:
":? "_PRINTER _"
100 TRAP 110:CLOSE #N3:OPEN #N3,N12,NO
,"R1":? "_ RS232 PORT #1_"
110 TRAP 120:CLOSE #N3:OPEN #N3,N12,NO
,"R2":? "_ RS232 PORT #2_"
120 TRAP 130:CLOSE #N3:OPEN #N3,N12,NO
,"R3":? "_ RS232 PORT #3_"
130 TRAP 140:CLOSE #N3:OPEN #N3,N12,NO
,"R4":? "_ RS232-C PORT _"
140 CLOSE #3:? W$(13):POKE 82,2:POKE 7
52,0:?
```



ATARI® DV

KISMET II



by Peter Kirsch

Kismet II is a dice game for the ATARI®, requiring 32K RAM, at least one joystick, and disk drive. It is included as a bonus program on this month's ATARI® Disk Version.

Game Description

The object of this game is to roll five dice to achieve certain combinations, thereby accumulating points on a score sheet. On each round, a player first rolls all five dice, and then has the option of rerolling any or all of them two additional times. Having achieved the best roll possible in this way, the player then must score the roll under one of 15 categories. These categories are grouped into an upper and a lower section on the score sheet.

The upper section has six categories, "Aces" through "Sixes." If the "Aces" category is chosen, the player receives one point for each die with a single dot up; if the "Twos" category is chosen, two points are received for each two rolled; and so on. The maximum possible scores for each of these six categories, then, are 5, 10, 15, 20, 25, and 30. A perfect score for this upper section would be 105.

The lower section has nine additional categories, which score as follows:

Two Pair, same color.....	total of dice
Three of a Kind.....	total of dice
Straight (1-5 or 2-6).....	30
Flush (all same color).....	35
Full House (pair + 3/kind).....	total + 15
Full House, same color.....	total + 20
Four of a Kind.....	total + 25
Yarborough (anything).....	total
Kismet (5 of a kind).....	total + 50



In order to receive the score indicated, the dice must contain the correct combination of numbers and colors.

Every roll must be scored in some category. This will sometimes mean being forced to take a zero score in one or more categories, for unusable rolls. It's to your benefit to make good on the upper section, since you receive bonus points for high totals: 35 bonus points added to a column total of 63-70; 55 added to a total of 71-77; and 75 to a total of 78 or more.

This version of *Kismet* enables each player to play four score sheets at a time, rather than just one. This method of play gives you more chances to play any given roll, and can make the game more interesting. Any roll may be played in any of the four columns.

Description of Play

The joysticks are used for all input once the game is under way. Toggling the stick up, down, left, or right will move the screen cursor in the corresponding direction. Toggling it diagonally down will switch between the display of the upper and the lower sections of the score sheet. The fire button is used to throw the dice and to record scores in the proper categories.

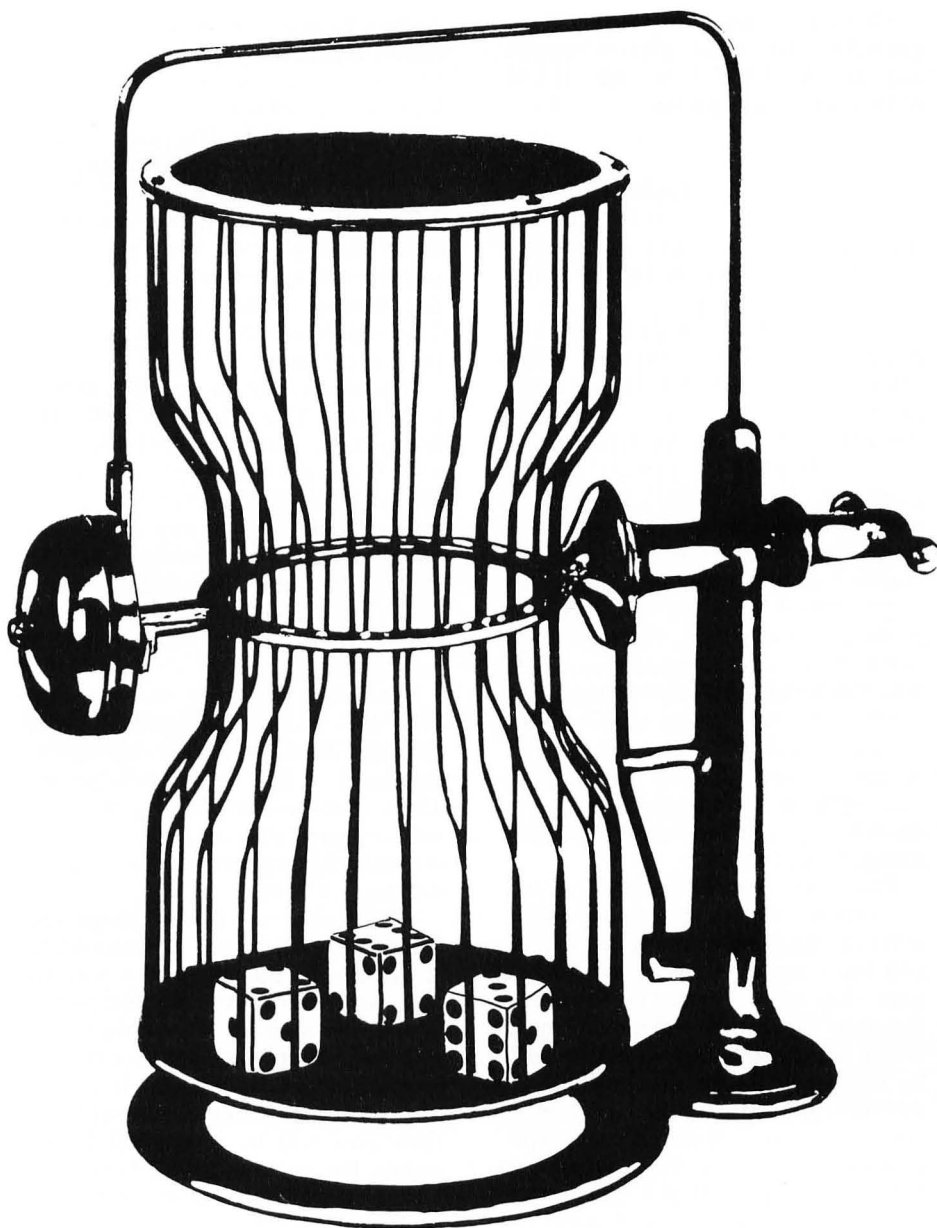
To throw the dice, position the green cursor in front of the black letter T. The black number indicates the current throw number; when it reaches zero the roll must be scored. To throw all five dice at once, just press fire. To reroll only certain dice, position the cursor in front of each number to reroll and press fire; a buzz will confirm your selection. Then, reposition the cursor in front of the T by toggling the stick up, and press fire to throw those dice you have selected.

To score a roll, position the cursor on the chosen category box and press fire. Remember that pushing the stick diagonally down will display the other half of the score

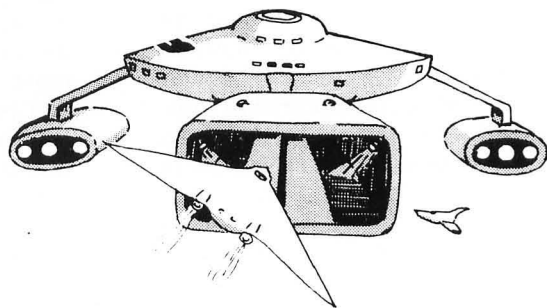
sheet. A roll which cannot be used in any of the unfilled categories must be scored anyway. To do this, move the cursor to the chosen box and press fire four times; after the fourth buzz, a zero will be entered in that category.

It is to a player's advantage to throw as many Kismets as possible. For each Kismet after the first, a

100-point bonus is awarded, no matter where it is actually scored — unless a zero has been entered in the Kismet box. A player's score sheet will be pink until he or she is in the bonus situation, and then it will turn turquoise. In addition, whenever a Kismet is rolled, a verse of *Happy Days Are Here Again* is played, amidst flashing lights and colors. 5



DEFENSE



by Greg Schroeder

Defense is an arcade-style program for one or two players requiring an ATARI® with 16K RAM (32K with disk) and one joystick.

The object of *Defense* is to destroy as many aliens that fly across the moon's surface as possible, before your three fighter ships have been destroyed.

When the game begins, you are given a brief description of your situation and mission while the computer redefines the ATARI® character set. Then you type 1 or 2 for the number of players in the game. A "get ready" prompt is given at the start of each player's turn. The computer will draw a text graphics display using the characters it previously defined in graphics mode 1, showing at the top each player's score, the highest score so far, and number of ships left; and at the bottom, how many aliens in the current attack wave remain. In the center is your playing field, with your special text graphics ship on the left side and the moon's surface along the bottom.

Player #1's joystick is plugged into socket #2, and player #2's joystick plugged into socket #3. If you push the joystick up or down, your ship will be raised or lowered. But don't go too far down because if you hit the ground, your ship will be destroyed. Push the joystick to the right and after about a second, you will gain airspeed and the ground will start to move. Pushing the stick to the left will cause you to lose airspeed and the ground to slow also.

As many as three aliens at a time will appear on the screen when they have reached your section of the moon's surface. The scoring system for each alien is as follows:

Yellow Drone: 10 points.
Green Drone: 20 points.
Blue Drone: 30 points.
Red Drone: 40 points.
Red Smiling Blob: 100 points.
Green Smiling Blob: 200 points.

To destroy these aliens, you must reach their altitude with your ship and fire a laser beam by pressing the fire button. When you hit them, their scores are shown and added to your total score.

The aliens fly in specific attack patterns. The drones will fly diagonally, bouncing off the top and bottom of the playing field, while the smiling blobs fly in an erratic pattern towards you and have a better chance of hitting you. Be careful because the smiling blobs begin in the center of the screen, unlike the drones which start at the far right.

If an alien does hit you, your ship will be shown exploding in a graphics display and one ship will be deducted from your possession. Then, if it is a one-player game, you will continue in a different ship, or if a two-player game, the next player will have a turn.

When you run out of ships the game is over for you and your score is displayed. But there is a way to gain extra ships. As you destroy the aliens, the number of remaining attackers in that wave will decrease on the bottom of the screen. If all the aliens in that wave are destroyed, then you will be given 500 bonus points for each ship you have left. Every time you clear out four such waves, you are awarded a bonus ship.

Note: The system reset key must be pressed before each RUN of the program to restore the ATARI® character set. For more information on redefining the ATARI®

character set, see *Take Apart: ATARI® Quest* and *Character Generator* in the October issue of *SoftSide*.

Variables

A(n,n): Nonvisible position of aliens along the moon's surface.
A, I, J, X,: Miscellaneous variables.
AL(n): Number of aliens in each player's current attack wave.
A\$: Moon surface picture string.
B(n), C(n): X and Y positions of aliens on screen.
D(n): Y movement of alien on screen.
E(n): Type of alien on screen.
F(1), F(2): Number of aliens in each player's attack wave. Used to reset AL(n).
F(3), F(4): Number of attack waves destroyed by each player.
GS: Ground speed. Controls speed of ground picture movement and alien movement.
H\$: Temporary string storage used in moving the moon's surface.
PL: Current player number.
PP: Total number of players.
R(n): Stores number of aliens along moon's surface that are on the screen.
S: Joystick value.
SC(n): Each player's score.
SH(n): Remaining ships for each player.
SL: Trigger pressed identifier. (1 = Trigger still pressed.)
ST: Beginning of redefined character set pointer.
TT: Counter for alien movement on screen.
XC: Counter for alien movement along moon's surface.
Y: Y axis of position of player's ship. (X defaults to 1.)
Y1: Temporary storage for Y.

ATARI®

```

#####
$ Atari BASIC $
$ 'DEFENSE' $
$AUTHOR: Greg Schroeder$
$ (c) 1982 SoftSide $
#####

```

Lines 10-50: Initialization

Jump to subroutine for title and character redefinition.

```
10 GOSUB 10000
```

Dimension variables.

```
15 DIM A(2,4),B(4),C(4),D(4),E(4),F(4),
SH(2),SC(2),A$(21),H$(5),R(4),AL(2):
SC(1)=0:SC(2)=0
```

Set arrays to zero.

```
20 FOR I=0 TO 4:B(I)=-1:C(I)=0:D(I)=0:
E(I)=0:R(I)=0:NEXT I
25 I=SC(1)*(SC(1)>SC(2))+SC(2)*(SC(2)>
SC(1)):HSC=HSC*(HSC>I)+I*(I>HSC)
```

Place aliens for two players in moon's surface variable.

```
30 FOR I=1 TO 2:FOR J=1 TO 20:A(I,J)=I
NT(RND(0)*16):NEXT J:NEXT I
```

Define other variables for both players.

```
40 SH(1)=3:SH(2)=3:SC(1)=0:SC(2)=0:AL(
1)=20:AL(2)=20:F(1)=20:F(2)=20:F(3)=0:
F(4)=0:XC=0
```

Jump to "get ready" subroutine.

```
50 GOSUB 6000
```

Lines 100-150: Main loop.

Increment alien on screen counter.

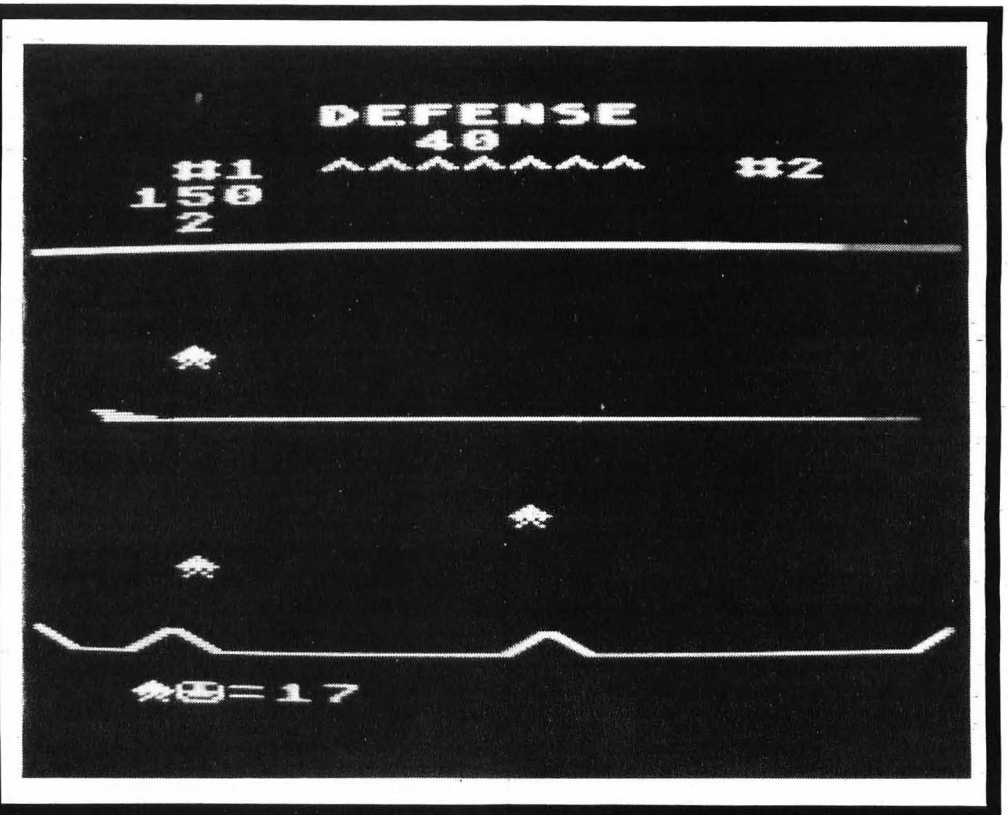
```
100 TT=TT+1:IF TT=4 THEN TT=1
```

Move moon's surface contained in A\$. Use H\$ as temporary storage.

```
105 IF GS>1 THEN H$=A$(1,GS):A$=A$(GS+
1,20):A$(LEN(A$)+1)=H$:POSITION 0,20:?
#6:A$;
```

Lines 110-140: Move Ship.
Check if joystick moved.

```
110 S=STICK(PL):IF S=15 THEN 150
```



Check joystick to move ship up.

```
120 Y1=Y:IF S/2=INT(S/2) AND Y>6 THEN
Y=Y-1
```

Check joystick to move ship down. See if ship hit ground.

```
125 IF (S=5 OR S=9 OR S=13) THEN Y=Y+1
:IF Y=20 THEN GOTO 3000
```

Erase and reprint redefined ship character.

```
130 POSITION 1,Y1:? #6;" ";:POSITION
1,Y:? #6;"XY";
```

Check joystick to move ship faster.

```
135 IF S>4 AND S<8 AND GS<2 THEN GS=GS
+0.05
```

Check joystick to move ship slower.

```
140 IF S>8 AND S<12 THEN GS=GS-0.05:IF
GS<0 THEN GS=0
```

Lines 150-170: Fire Laser.

Check if trigger released.

```
150 IF STRIG(PL)=1 THEN SL=0:GOTO 200
```

If trigger still pressed, then branch to alien movement.

```
155 IF SL=1 THEN 200
```

Print redefined text graphics laser. Make some noise. Check if laser hit alien(s).

```
160 SOUND 1,20,4,15:POSITION 3,Y:? #6;
"_____";:FOR I=1 TO 3:IF B(
I)>2 AND INT(C(I))=Y THEN 4000
161 REM LINE 160: 16 <CTRL> '._'
162 NEXT I
```

Erase laser. Turn off sound. Set trigger pressed variable.

```
165 POSITION 3,Y:? #6;"
";:SOUND 1,0,0,0:SOUND 2,0,0,0:SL=1
```

Check if all aliens are gone.

```
170 IF AL(PL)<1 THEN 4100
```

Lines 200-300: Alien Movement.
Increment moon surface counter.

```
200 XC=XC+1:IF XC=F(PL)+1 THEN XC=1
```

Move aliens' position in array.

```
210 IF A(PL,XC)>-5 THEN A(PL,XC)=A(PL,
XC)-(GS):IF A(PL,XC)<=0 THEN 1000
```

Check if variable contains no alien.

```
220 IF B(TT)=-1 THEN 300
```

Erase alien. Move alien along X axis. Check if it is off-screen.

```
230 POSITION INT(B(TT)),INT(C(TT)):? #
6;" ";B(TT)=B(TT)-65-0.5:IF B(TT)<0 T
HEN B(TT)=-1:A(PL,R(TT))=15:GOTO 300
```

Check if alien is smiling blob and branch to different movement routine.

```
240 IF E(TT)>4 THEN 2000
```

Check if alien is off-screen after moving along Y axis.

```
250 C(TT)=C(TT)+D(TT):IF C(TT)<6 OR C(
TT)>19 THEN C(TT)=C(TT)-D(TT):D(TT)=-D
(TT)
```

Check if alien hit you.

```
255 IF INT(B(TT))>=0 AND INT(B(TT))<=2
THEN IF INT(C(TT))=Y THEN 3000
```

Print specific redefined alien character.

```
260 POSITION INT(B(TT)),INT(C(TT)):GOS
UB 260+E(TT)#5:GOTO 300
```

```
265 ? #6;"z";:RETURN
```

```
270 ? #6;"z";:RETURN
```

```
275 ? #6;"z";:RETURN
```

```
280 ? #6;"z";:RETURN
```

```
285 ? #6;"t";:RETURN
```

```
290 ? #6;"t";:RETURN
```

Return to start of main loop.

```
300 GOTO 100
```

Lines 1000-2010: Set Up New Alien

Find empty alien variable: If none, then restart alien at end of moon's surface.

```
1000 FOR I=1 TO 3:IF B(I)=-1 THEN 1010
1005 NEXT I:A(PL,XC)=15:GOTO 220
```

Set Y value for alien. Set alien's movement.

```
1010 C(I)=INT(RND(0)#12)+7:D(I)=RND(0)
+0.1-1.5*(RND(0)<0.5)
```

Pick alien type. Remember which surface alien it is. If alien is a smiling blob, then start X position at mid-screen.

```
1020 E(I)=INT(RND(0)#6)+1:R(I)=XC:A(PL
,XC)=-10:B(I)=17:IF E(I)>4 THEN B(I)=1
1
```

```
1025 GOTO 220
```

Move smiling blob along Y axis.

```
2000 IF C(TT)<Y THEN D(TT)=1:GOTO 250
2010 IF C(TT)>Y THEN D(TT)=-1:GOTO 25
0
```

Lines 3000-3070: You've Been Hit. Clear playing field.

```
3000 FOR I=6 TO 19:POSITION 0,I:? #6;"
";:NEXT I
```

Reset screen aliens to surface start.

```
3010 FOR I=1 TO 3:IF B(I)>0 THEN A(PL,
R(I))=15:B(I)=-1
```

Turn off sound.

```
3015 NEXT I:FOR I=0 TO 3:SOUND I,0,0,0
:NEXT I
```

Print redefined ship character. Brighten ship in red.

```
3020 POSITION 1,Y:? #6;"XY";:FOR I=2 T
O 8 STEP 0.1:SETCOLOR 2,4,I:NEXT I
```

Make explosion sound, first explosion. Initialize explosion loop.

```
3025 SOUND 0,20,8,15:POSITION 1,Y:? #6
;"##";:FOR I=15 TO 0 STEP -0.2
```

Make explosion sounds.

```
3030 SOUND 0,20,8,1:SOUND 1,75,8,I+1:S
OUND 2,175,8,I+3:SOUND 3,255,8,I+5
```

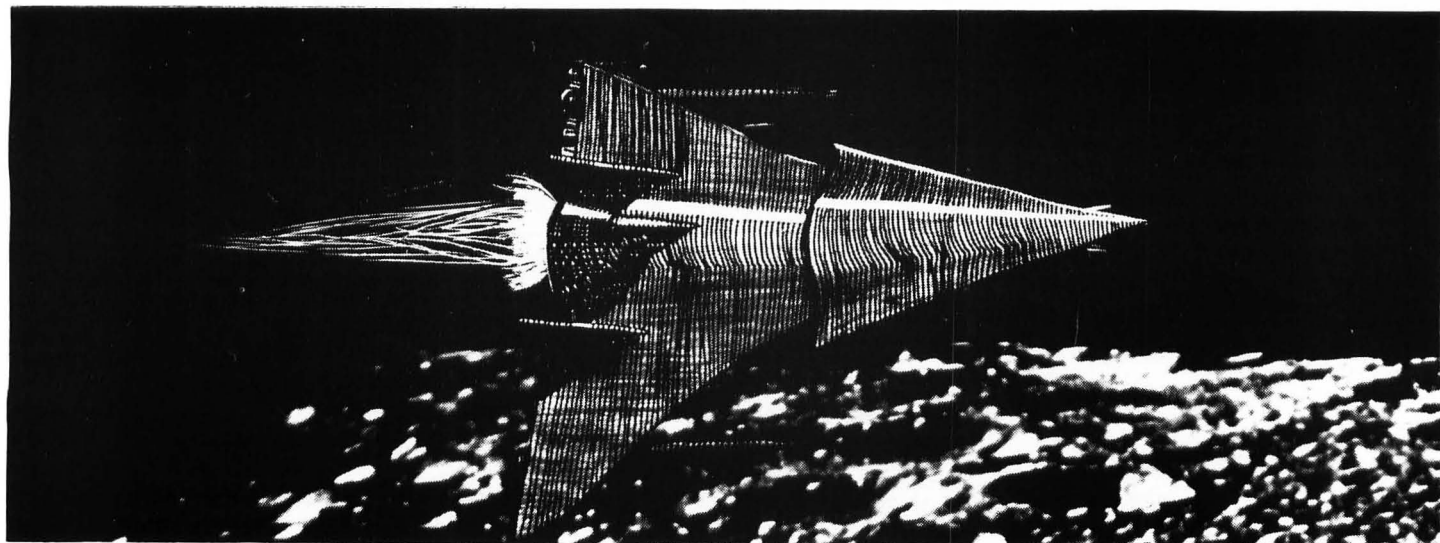
Print explosion steps when ready.

```
3040 IF I=12 THEN POSITION 0,Y-1:? #6;
"###";:POSITION 0,Y:? #6;"_:_:";:POSIT
ION 1,Y+1:? #6;"##";
```

```
3045 IF I=8 THEN POSITION 0,Y-1:? #6;"
_:_:";:POSITION 0,Y:? #6;"_+_:";:POSIT
ION 0,Y+1:? #6;"_:_:";
```

```
3050 IF I=4 THEN POSITION 0,Y-1:? #6;"
+_:";:POSITION 0,Y:? #6;"_+_:";:POSIT
ION 1,Y+1:? #6;"_:";
```

```
3055 IF I=1 THEN POSITION 0,Y-1:? #6;"
_:";:POSITION 0,Y:? #6;"_:";:POSIT
ION 0,Y+1:? #6;"_:";
```



Turn off sound. Erase ship.

```
3060 NEXT I:FOR I=0 TO 3:SOUND I,0,0,0
:NEXT I:POSITION 1,Y:? #6;" "
```

Set ground speed to zero. Subtract ship from player. Print number of ships left. Check if all ships gone.

```
3062 GS=0:SH(PL)=SH(PL)-1:POSITION 3+1
2*(PL=2),4:? #6;SH(PL);IF SH(PL)=0 TH
EN GOSUB 7000
```

If there's another player, then change player variable.

```
3065 IF SH(PL-1+2*(PL=1))>0 THEN PL=PL
+1:IF PL>PP THEN PL=1
```

Jump to "get ready" routine. Jump to screen drawing.

```
3070 GOSUB 5000:FOR I=1 TO 1000:NEXT I
:GOSUB 6000:XC=0:GOTO 100
```

Lines 4000-4050: Destroy Aliens.

Make some noise. Check if alien is smiling blob.

```
4000 SOUND 2,RND(0)*200+50,10,10:POSIT
ION INT(B(I)),INT(C(I)):? #6;:IF E(I)>
4 THEN 4020
```

Print score for drone. Add to total score.

```
4010 ? #6;E(I)*10;:SC(PL)=SC(PL)+E(I)*
10:GOTO 4030
```

Print score for smiling blob. Add to total score.

```
4020 ? #6;(E(I)-4)*100;:SC(PL)=SC(PL)+
(E(I)-4)*100
```

Clear alien variables. Print player's score.

```
4030 B(I)=-1:A(PL,R(I))=-5:POSITION 2,
3:? #6;SC(1);:IF PP=2 THEN POSITION 14
,3:? #6;SC(2);
```

Decrease the number of aliens in attack wave. Return to see if more hit.

```
4040 AL(PL)=AL(PL)-1:POSITION 2,22:? #
6;"zt=";AL(1);" ";:IF PP=2 THEN POSITI
ON 12,22:? #6;"zt=";AL(2);" ";
4050 NEXT I:GOTO 165
```

Lines 4100-4135: Attack Wave Destroyed.

Clear screen. Restore ATARI® character set. Add to player's wave cleared total. Print message, make noise.

```
4100 GRAPHICS 17:POKE 756,224:F(PL+2)=
F(PL+2)+1:POSITION 8,10:? #6;"attack":
? #6;" wave ";F(PL+2);
4110 ? #6;" destroyed":? #6:SOUND 0,20
0,10,10:SOUND 1,201,10,10:SOUND 2,0,0,
0
```

Print bonus message. (500 points for each remaining ship.) Add bonus to total score.

```
4120 ? #6;" BONUS - ";SH(PL)*500:SC
(PL)=SC(PL)+SH(PL)*500
```

Check if extra ship is awarded. Print message.

```
4123 IF F(PL+2)/4=INT(F(PL+2)/4) THEN
SH(PL)=SH(PL)+1:? #6:? #6;" extra 5
hip"
```

Add more to next wave. Reset aliens on surface.

```
4125 AL(PL)=F(PL)+5:F(PL)=F(PL)+5:IF F
(PL)>40 THEN F(PL)=40:AL(PL)=40
```

Make sure wave doesn't go over 40 aliens.

```
4130 FOR J=1 TO AL(PL):A(PL,J)=INT(RND
(0)*16):NEXT J
```

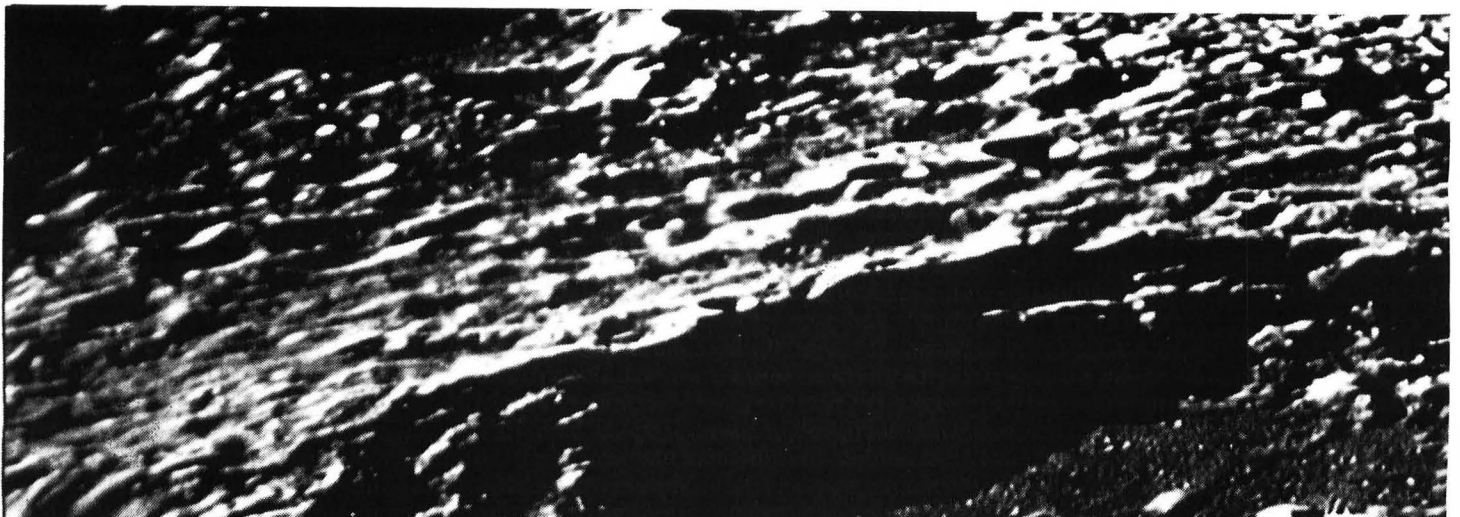
Jump to screen drawing. Jump to main loop.

```
4135 FOR I=1 TO 1500:NEXT I:GOSUB 6000
:GOTO 100
```

Lines 5000-5005: Get Ready

Clear screen. Restore ATARI® character set. Print message.

```
5000 GRAPHICS 18:POKE 756,224:POSITION
4,5:? #6;"PLAYER ";PL
5005 POSITION 4,6:? #6;"GET READY!!":R
ETURN
```



Lines 6000-6070: Draw Screen.

Clear screen. Restore new character set.
Turn off sound.

```
6000 GRAPHICS 17:POKE 756,ST/256:FOR X
=0 TO 3:SOUND X,0,0,0:NEXT X
```

Print messages for both players. (If only one player, then only player #1's messages.)

```
6020 POSITION 6,0:? #6;"defense":POSIT
ION 3,2:? #6;"#1 ^^^^^^ #2":POSITION
(18-LEN(STR$(HSC)))/2,1:? #6:HSC;
6030 POSITION 2,3:? #6;SC(1);:POSITION
3,4:? #6;SH(1);
6040 IF PP=2 THEN POSITION 14,3:? #6;S
C(2);:POSITION 15,4:? #6;SH(2);
6045 POSITION 2,22:? #6;"zt=";AL(1);:I
F PP=2 THEN POSITION 12,22:? #6;"zt=";
AL(2);
```

Print line made of redefined character.

```
6050 FOR I=0 TO 19:POSITION I,5:? #6;"
W";:NEXT I
```

Assemble moon's surface picture in A\$.

```
6060 FOR I=1 TO 20:A$(I,I)=" ":NEXT I;
A$(10,11)="UV":A$(2,3)="UV":A$(19,20)=
"UV"
```

Print ground. Set ship position (Y) to mid-screen and print ship. Make rocket sound.

```
6070 POSITION 0,20:? #6;A$;:Y=12:POSIT
ION 1,Y:? #6;"XY";:SOUND 0,75,8,4:RETU
RN
```

Lines 7000-7050: Game Over.

Restore ATARI® character set. Print game over and score.

```
7000 POKE 756,224:GRAPHICS 18:POSITION
5,5:? #6;"PLAYER ";CHR$(PL+176);:POSI
TION 5,6:? #6;"game over";
7005 POSITION 4,8:? #6;"score = ";SC(P
L);
```

See if both players' games are over.

```
7010 FOR I=1 TO 400:NEXT I:IF SH(1)=0
```

```
AND SH(PP)=0 THEN 7050
```

Return to main loop.

```
7020 RETURN
```

POP last GOSUB from stack. Jump to title routine. Jump to number of players input. GOTO setup of variables.

```
7050 FOR I=1 TO 400:NEXT I:POP :GOSUB
11000:GOSUB 11100:GOTO 20
```

Lines 10000-10070: Redefine Character Set.

Set top of memory five pages down. Initialize graphics. Jump to title display routine.

```
10000 POKE 106,PEEK(106)-5:GRAPHICS 0:
GOSUB 11000
```

Set start of new character set pointer.

```
10010 ST=(PEEK(106)+1)*256
```

Move ATARI® character set from ROM to top of RAM. Jump to number of players input.

```
10020 FOR X=0 TO 1023:POKE ST+X,PEEK(5
7344+X):NEXT X:GOSUB 11100
```

Redefine "X" character to be back half of ship and "Y" to be front half of ship. "Z" is a drone. "U" is CHR\$(6). "V" is CHR\$(7).

```
10030 FOR X=0 TO 23:READ A:POKE ST+448
+X,A:NEXT X:FOR X=0 TO 15:POKE ST+424+
X,PEEK(ST+560+X):NEXT X
```

"W" is CHR\$(18).

```
10040 FOR X=0 TO 7:POKE ST+440+X,PEEK
ST+656+X):NEXT X
```

"T" is smiling blob.

```
10045 FOR X=0 TO 7:READ A:POKE ST+416+
X,A:NEXT X
```

"@" is flat ground and laser character.

```
10050 FOR X=0 TO 6:POKE ST+256+X,0:NEX
T X:POKE ST+263,255
```

Initialize new character set.

```
10055 GRAPHICS 1:POKE 756,ST/256:RETUR
N
```

DATA for redefined characters.

```
10060 DATA 0,0,0,0,30,15,7,7,0,0,0,0,0
,192,252,255,0,8,28,62,127,42,20,34
10070 DATA 0,126,153,153,255,189,195,1
26
```

Lines 11000-11050: Title Display.

Suppress cursor. Print title and instructions.

```
11000 GRAPHICS 17:POKE 752,1:POSITION
7,0:? #6;"defense":? #6:? #6;" e
vil aliens from"
11010 ? #6;" beyond earth have invad
ed the moon.":? #6:? #6
11020 ? #6;" YOUR MISSION IS TO DE
STROY AS MANY ATTACKING WAVES OF THO
SE ALIENS BEFORE";
11030 ? #6;" THEY LAUNCH THEIR FLEET
S TO CONQUER THE EARTH.":? #6:? #6
11040 ? #6;" YOUR JOYSTICK CONTR
OLS YOUR SPEED AND ALTITUDE. PUSH THE
BUTTON TO FIRE."
11050 RETURN
```

Lines 11100-11120: Number of Players Input.

OPEN keyboard. Print message.

```
11100 OPEN #2,4,0,"K":POSITION 0,23:?
#6;"PUSH 1 OR 2 TO START";
```

Check if key "1" or "2" is pressed.

```
11110 IF PEEK(764)<>30 AND PEEK(764)<>
31 THEN 11110
```

GET key pressed. CLOSE keyboard. Decide whether one- or two-player game.

```
11115 GET #2,I:PP=1:CLOSE #2:IF I=50 T
HEN PP=2
```

Set player variable. Jump to "get ready" subroutine.

```
11120 PL=1:GOSUB 5000:RETURN
```


What's New

Ramdisk

From AXLON Inc. System requirements: 24K ATARI® 800. Projected retail price: \$450.

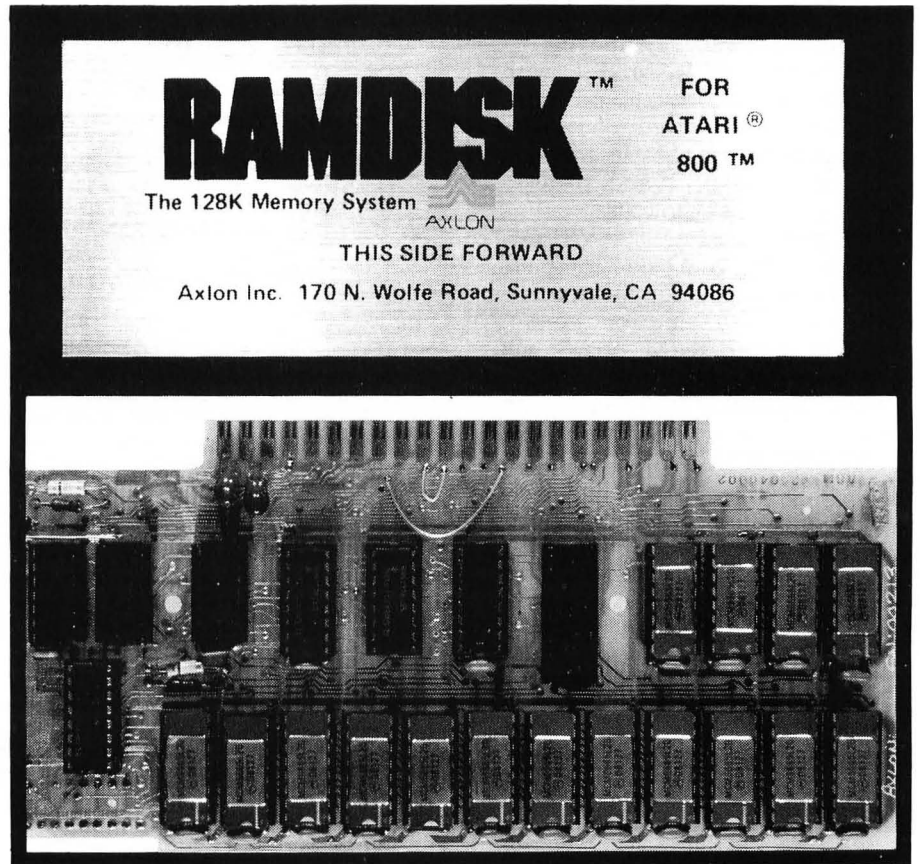
Reviewed by Dean F.H. Macy

Several months ago I mentioned a wristwatch-sized computer complete with terminal and microscopic disk drive as an indication of things to come in the far future. Sitting here, holding a 6x3x.5 inch disk drive with no moving parts in the palm of my hand, I suspect the future is not quite as far away as I earlier imagined.

The new drive, named "RAM-DISK" by its designers at AXLON Inc., is a product of high-quality engineering technology. Although it is not truly a disk drive the RAM-DISK out-performs any drive on the market, locating and displaying data in micro-micro seconds. What would take five seconds to save on a conventional disk takes a 25th of a second using RAMDISK.

Thanks to readily obtainable support from AXLON, an informative 14-page user's manual and supplied software disk, the RAMDISK is a joy to use. RAMDISK is inserted into ATARI®'s memory-slot three where it is flanked by two 16K memory cards giving you 160K bytes of fully addressable user RAM.

What can you do with all this RAM? If you're into graphics and use 32K RAM for programming, you still have 131,072 bytes in 16 separate areas immediately available at the same address. This means you can instantly switch to 16 different screens faster and easier than altering the display list. Since it is the same memory area, switching segments puts different information on the screen. You can also switch in the middle of a screen, seeing part of your screen from one area, and part from another. Looking at all this RAM from another viewpoint, just think what you could do with 131 graphics mode 0 screens, 255 graphics 1 screens, 502 graphics 2 screens, or 33 graphics 7 screens



loaded in and ready to go. Now that's exciting!

The greater part of RAMDISK users will allow RAMDISK to be their second disk drive. For this purpose, AXLON supplies a disk to be copied to your own ATARI® 2.0 DOS creating the RAMDISK Memory Management Software. This software contains documentation for the RAMDISK as well as "NOMEMSAV," "ASSIGN," and "CREATE" programs; the "AUTORUN.SYS" program; a "RAMSCAN" diagnostic and a "MEMTEST" program.

I'm sure ATARI® owners have been frustrated with the extremely slow process of making disk backups. (With one drive a normal backup takes 13 minutes. With two, backups still take over nine minutes.) Using RAMDISK, *SoftSide* has been making ATARI® disks in under three minutes, in-

cluding formatting. This is possible because all the data on one disk can be loaded into RAMDISK which serves as the source for making copies.

RAMDISK is a computer-etched, double-sided PC board, sandwiched between full-sized plastic insulating covers. The memory chips used are Motorola D8127s. All chips are inserted into mounted IC sockets and run lukewarm, even when used heavily over several hours. The memory support chips are standard buffer and routing ICs and have been trouble-free during the three weeks of *SoftSide* testing.

As a disk, RAMDISK is addressed like any other formatted disk. A directory of RAMDISK will yield the same information as a directory of a real disk. RAMDISK can be assigned the designation "D1" thereby allowing full access of disk data. Using RAMDISK as a

ATARI®

POKE YOUR ATARI®

disk does have some drawbacks. For instance, RAMDISK cannot hold data indefinitely like a real disk. If the power goes off RAMDISK goes off too and all memory is erased. Although this is easily ratified by making backups on real disks, power outages can be annoying. (I would like to see RAMDISK with temporary battery backup power.)

AXLON has given ATARI® microcomputers a shot in the arm with the RAMDISK and plans a similar event for Apple users. During the early part of 1982, AXLON plans a 320K ramdisk, complete with its own power supply and battery backup, available for the Apple. The Apple RAM board will serve as an additional 320K memory or two, completely addressable disk drives utilizing similar features of the ATARI® RAMDISK. (No prices yet.)

After weighing the positives and negatives between RAMDISK and a second ATARI® disk drive I would definitely purchase AXLON's RAMDISK if I owned an ATARI® system.

Part III: The video screen.

Atari Inc. has put a lot of work into the hardware of the ATARI® 800 and 400. Contained within the plastic shell which forms the cover and surrounds the keyboard, a multitude of custom-made integrated circuits carry on many useful and/or special tasks. Items such as player/missile graphics, the display list, custom character sets, sound, and color are the work of many hours of engineering and design.

This month's POKE is a simple yet interesting one, especially for those who have word processors or adventure programs. Almost everyone knows that you can get inverse video by pressing the ATARI® logo key prior to typing another key. However, this can be quite bothersome if there is a lot of material to be printed in this form. Also the background color makes the inverse text seem highlighted: a nice thing to have if you want it.

But stop and think for a minute. When you look at a printout or a piece of paper with writing on it, the letters are black on a white background. Now look at the display of your ATARI®: white letters on blue. Case studies have shown that black on white tends to reduce eyestrain, as well as make words easier and faster to read.

So we want inverse video, but the BASIC interpreter will not accept any command but LIST and RUN in inverse. The way to get around this is to tell the hardware that the colors of the text and the background on the screen are to be different than they are when the computer is turned on.

Since almost every important feature of the ATARI® has a corresponding "shadow-register" in memory, all we need are the locations of the text color register and the background color register. These locations are 709 and 710 respectively. In a color register, 0 to 15 means the same as black to white. So to get a very good combination of black on white, try this:

POKE 709,4: POKE 710,15

This gives inverse all the time, with the ATARI® logo key producing normal text. Note that a GRAPHICS 0 command will reset the colors — use PRINT CHR\$(125). It also clears the screen, but it won't do nasty things like reset all color registers to default values, reset the character set pointer to the standard set in ROM, etc. The same thing can be accomplished with:

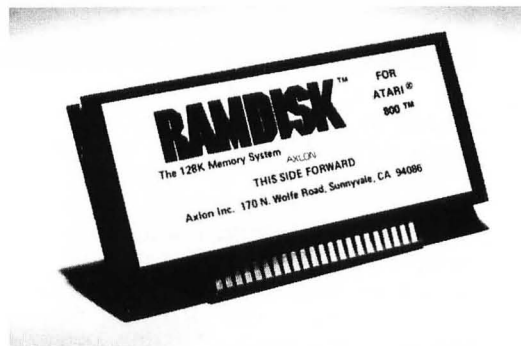
SETCOLOR 1,0,4: SETCOLOR 2,0,15

but the POKES take fewer bytes and after all — this is called *POKE Your ATARI®*, not *SET-COLOR Your ATARI®*.

So now you can modify (if you care to) *Microtext*, BASIC adventures, and anything that deals with a large amount of text. Until next month — enjoy, and don't forget to POKE YOUR ATARI!

Alan J. Zett
SoftSide

ATARI* 800* OWNERS!



PLUG IN AND GO !

The Axlon RAMDISK Memory System provides 128K of RAM memory which can be utilized as an additional disk device or bank selectable RAM memory! The DOS supplied with the system allows you to utilize the RAMDISK Memory System as a disk device in conjunction with your Atari 810*. The system is up to 20 times faster than the Atari 810 and is compatible with existing Atari 800 software. As user memory, the RAMDISK Memory System is organized as eight (8) 16K banks. The system is installed with two 16K RAM modules giving you a 160K Atari 800 system. Drop by your local computer store for a demonstration or contact Axlon, Inc. for more information.

- Plug-in Compatibility
- 128K Bytes of RAM Memory
- Compatible with existing Atari 800 Software
- Can be utilized as an additional disk - function for function, up to 20 times faster than the Atari 800
- Includes DOS Memory Management Software
- Can be utilized as Bank Selectable User Memory
- Gold Plated Contacts
- 90 Day Warranty

* Indicates Trademark of Atari, Inc.

170 N. Wolfe Road
Sunnyvale, CA 94086
(408) 730-0216



Protector

A review by Alan J. Zett

By Mike Potter (Synapse Software)
System requirements: 32K
ATARI® with cassette or disk and
one joystick. Suggested retail price:
tape — \$24.95, disk — \$27.50.

Imagine a world where UFOs kidnap people, aliens roam around causing havoc, burning lava can totally destroy an entire city, and meteors fall from the sky. Sound familiar?

Be that as it may, the world I'm talking about is the world of *Protector*, a new real-time arcade-type Machine Language program from Synapse.

Protector is not just another "I-shoot-em-up-and-they-die" arcade game. Your mission, should you decide to accept it, is to play the cosmic hero and rescue the natural inhabitants of an unknown planet from the clutches of the hostile aliens who are attempting to obliterate them. But don't be fooled into thinking the aliens will sit there and let you!

You start at a refueling station with a sturdy airship and the knowledge that only 18 people remain alive. The multi-colored landscape stretches out before you with many obstacles to overcome. In fact, the landscape is so large that only a portion of it is visible at any one time. There are mountains and buildings for you to accidentally fly into, laser posts attempting to fry you, meteors trying to smash you and aliens trying to ram you or eat the people you carry.

There are two main phases to playing the game (however, as you play it many more phases become apparent, depending on your current playing strategy). First you must move as many people as you can from the left-hand city to the right. This is done by flying close to the person you wish to carry. When you've caught him, you'll see his arm drop to signal you that he has a

hold on the ship. Then you can lift off, jerkily pulling him across the screen and he will stay attached until you fire your laser. Then he drops down to the ground. Throughout all this, a UFO (which is impervious to laser fire) is hindering you by slowly grabbing people one by one with a tractor beam (which you can accidentally run into as I've discovered many, many times) and taking them over to the volcano for a proper frying. If you're fast enough and careful enough, you can attempt to save the people as they are dropped in, but remember, there are only 18 survivors and your ship can be refueled to carry on the mission.

After the remaining people are safely resting in the right-hand city, the volcano then decides that the time is right to erupt and lava starts flowing towards the right-hand city. Now you must transfer people to the only other safe place on the planet: the pneumatic tubes. But watch out! To get there, you must cross a ground-based laser station, a landscape bathed in a meteor storm, and finally, right above the tubes, another intricate weave of laser posts. This game is one that does not let you rest!

Scoring is based on points for destroying aliens, meteors, and laser posts; the number of people moved from the left city to the right; and the number of people deposited in the pneumatic tubes.

At the very start of the game, the ATARI® plays some dramatic music to get you into the mood and lets you select from six different levels of play with a choice of three or five ships.

Levels 1, 2, and 3 are roughly equivalent to an easy, medium, and hard mode using an easy map (the map being the landscape). Also, in Level 1 no meteors fall, making it perfect for learning the game. Levels 4, 5, and 6 are also easy, medium, and hard levels, but they use a more complicated map. The

easy map gives you more room to fly about, while the harder map is cramped and contains more obstacles, giving you less room for mistakes.

One thing you learn quickly is that there is always a different way to play the game. After a few games with meager scores, a good study of the landscape and the way the game reacts will give you a better idea of how to play the next game. One thing is for sure, no one will ever totally master this game. I was having a hard time getting more than five people all the way to the end until someone pointed out to me that there are many helpful places that can be used temporarily when the time is right. (If I said any more, I'd give away one of many strategies I've developed and probably ruin the game for some poor struggling novice.)

From an ATARI® programmer's point of view, this game uses almost all of the ATARI®'s hidden features in a splendid display of machine-code programming. I really have to give Mike Potter a hand for the amount of detail he has put into this game. Two of the features I noticed immediately were: a well-done redefined character set that has the classic "computer-type" look to it and a modified display list for mixing text at the top of the screen and graphics in the center. The landscape itself is a huge map of Hi-Res graphics approximately ten screens wide. The map is moved using the smooth horizontal-scroll technique. And I'm sure that with all the action and color on the screen, that there are many player-missile graphics in use. The sound effects, color, graphics, and style are combined very well into what I can only call a high-quality piece of programming art. (In case anyone reading this is saying "What in the world is he talking about? What are player-missile graphics, etc.?", I suggest you purchase a copy of the book *De Re ATARI®* from Atari, or read

ATARI®

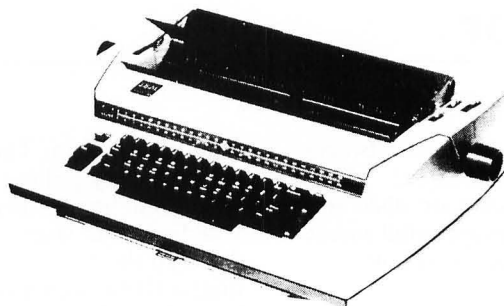
past issues of *SoftSide* and *Compute!* where you will find many explanations of the previously mentioned "Hidden ATARI® features.")

Normally, in my reviews, this paragraph is set aside to sum up any complaints I have about a program. For the first time, I cannot think of one thing that I would like changed or removed. However, another person who played the game did remark to me that "I don't think the player should die when he runs into a person; the person should die and that's it." This is nit-picking, at best. There just isn't anything major to gripe about.

While in the end, you must judge for yourself, my conclusion is that this program will be one of my favorites for a long time, right up there with *Star Raiders* and *Missile Command*. It is the first major piece of software for the ATARI® I've seen on the market that is a worthy rival to Atari's line of cartridge software. S

WHEN YOU SPEND SO MUCH FOR A PRINTER,
YOU SHOULD HAVE ONE THAT YOU CAN USE---
Introducing....

THE IBM TOTAL PRINTER/TYPEWRITER



FEATURES:

10 and 12 Pitch, Proportional Space, Full Typewriter Use, Auto Correcting, Sound Cover, "Smart" Keyboard

SPECIFICATIONS:

200 WPM Throughput, Either Serial OR Parellel, Self-Test, Lowest On-Site Maintenance, IBM Backed Printer. Cables stocked for all Apple, TRS(I, II, III), RS-232 systems.

PRICE: ONLY \$1995 (With 30 Day IBM Service Agreement)

CONTACT:

ICOM 11 N. Main, Lombard, Illinois 60148 (312) 932-1766

SPORTS FANS! MASTERS' GOLF WORLD SERIES

by David Bohlke

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

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

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

by David Bohlke

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

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

**SoftSide
Selections**

6 South Street Milford NH 03055

K-Byter

Lites Out

A TRS-80® K-Byter by Ronald and Jordan Corn, Den-
ville, NJ

Lites Out is a one-player arcade game with sound. It incorporates the sound routine which first appeared in the October, 1980, issue of *SoftSide*.

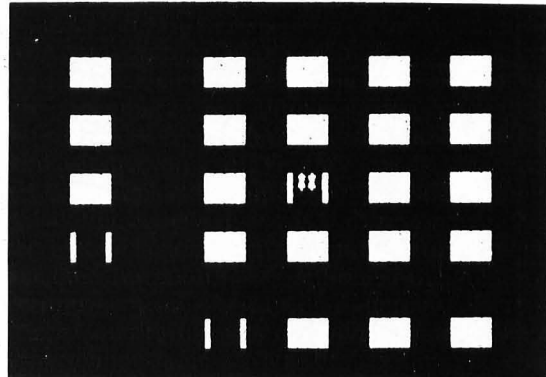
The object of the game is to turn off 16 graphics blocks representing "lights," which are grouped in a square arrangement at the top center of the screen. There are two cursors: one in a column of lights offset toward the left, and another in a row of lights offset toward the bottom. Pressing any key turns out the light at the row indicated by the left-hand cursor and the column indicated by the bottom cursor. In other words, the light turned out is at the "intersection" of the two cursors.

The time remaining in the game is indicated at the lower left-hand corner of the screen. It can be increased in two ways: by turning out the special bonus light (represented by a double asterisk, **), or by turning out all 16 lights. You will receive a severe penalty if you try to put out a light that is already out. Your score, displayed in the upper right-hand corner of the screen, depends on your speed and on your ability to turn off all the lights with no mistakes.

The game suggests a range of 20 to 200 for difficulty level, the lower levels being more difficult. Beginners should start with high numbers; accomplished experts may choose to defy the suggested limit of 20, and go for even smaller numbers (as low as 1).

The bottom cursor cycles from left to right. After two cycles of the bottom cursor, the left-hand cursor moves to a random location and the time remaining is decreased by 1. Each time a light is put out in one of the first three columns of a given row, the bottom cursor skips over the light in the next column. This makes it impossible to put out an entire row of lights in one cycle of the bottom cursor.

The bonus light appears randomly at the beginning of each set of 16 lights, and then disappears and reappears at random. The more lights remain lit, the more likely it is to appear. Putting it out adds 5 to your time remaining. Putting out all 16 lights adds 10 to your time remaining, and starts you with a new set of lights at a slightly harder level.



Your score is based on the number of lights out (N) and the time remaining (T). At any given moment, it is $(T + 1) * N$; therefore, it doesn't always increase. If you put all the lights out with ten seconds left, those extra seconds translate into a 160-point bonus. Each time you put out 16 lights, your score becomes "safe" — although it will still fluctuate, it will never go below this level.

```

5 MM$="EXACTLY TWENTY-SEVEN SPACES"
10 RANDOM:I=VARPTR(MM$):J=PEEK(I+1)+256*PEEK(I+2):FORK=JTDJ+26:R
EADX:POKEK,X:NEXT:POKE16526,PEEK(I+1):POKE16527,PEEK(I+2):DATA20
5,127,10,77,68,62,1,105,211,255,45,32,253,60,105,211,255,45,32,2
53,13,16,238,175,211,255,201
11 'FOR DISK SYSTEM CHANGE LINE 10 AS FOLLOWS:
...READX:POKEK,X:NEXT:DEFUSR0=J:DATA205,127,...
12 INPUT"DIFFICULTY (20-200)";L:T=25:F$=STRING$(4,191):B$=CHR$(1
49)+"**"+CHR$(170):E$=CHR$(149)+" "+CHR$(170)
15 C=0:B=1:D=RND(4)+1:E=RND(4):CLS:PRINT@54,S;R=0:FORY=0TO8STEP
2:R=R+1:FORX=0TO32STEP8:Q=Q+1:P(Q,R)=Y*64-64*(R=5)+X-5*(X<>0):PR
INT@P(Q,R),F$;S(Q,R)=1:NEXTX:Q=0:NEXTY:PRINT@P(1,5)," ";:PR
INT@P(D,E),B$;:PRINT@960,T;
150 Y=RND(4):PRINT@P(1,Y),E$;:FORN=1TO2:FORX=2TO5:PRINT@P(X,5),E
$;:M=USR(1000*(4+X)):FORZ=1TOL:NEXT:PRINT@P(X,5),F$;:IFINKEY$(<>
"GOTO250
160 IFRND(50)<>16GOTO230ELSEIFB=1PRINT@P(D,E),F$;
170 D=RND(4)+1:E=RND(4):IFS(D,E)=0B=0ELSEB=1:PRINT@P(D,E),B$;
230 NEXTX,N:PRINT@P(1,Y),F$;:M=USR(256)+USR(256)+USR(256):T=T-1;
PRINT@54,S+C*(T+1);:IFT<1PRINT@768,"GAME OVER!":FORZ=300TO750:M=
USR(Z):NEXT:ENDELSEPRINT@960,T;:GOTO150
250 IFS(X,Y)=0T=INT(T/5)+1:PRINT@960,T;:FORZ=300TO400:M=USR(Z):N
EXTZ:GOTO230
255 IFX=DANDY=ETHENB=0:M=USR(230):T=T+5:PRINT@960,T;
260 PRINT@P(X,Y)," ";:S(X,Y)=0:M=USR(9999):C=C+1:IFC=16T=T+10
:M=USR(0)+USR(0)ELSEX=X+1:GOTO230
270 S=S+C*(T+1):L=.9*L:X=9:NEXTX:GOTO15

```

5

Envyrn™

Putting
micro-
graphics
into
action



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

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

Envyrn™

Participation Program

6 South Street, Milford, NH 03055

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

Name _____

Address _____

City/State _____ Zip _____

Payment enclosed

MasterCard VISA Exp. Date _____

Name of Cardholder _____

MC# and Interbank#/VISA# _____

Signature _____

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

TRS-80 is a registered trademark of Tandy Corporation

SoftSide

TRS-80® DV


HELP PACKAGE

HELP

by Rich Bouchard

***HELP Package* is a TRS-80® disk utility. It is included as a bonus on this month's TRS-80® Disk Version.**

The HELP program is a utility for TRS-80® disk users that provides a quick, easy-to-use method of obtaining instructions on how to use features of your disk operating system, your computer itself, or anything else you wish to include. For example, if you needed to know what the syntax for the ATTRIB command was, you would type HELP ATTRIB from TRSDOS, and you would be shown. You could also print the instructions to a printer, using HELP\$ ATTRIB. Much more complete instructions on using HELP and on creating customized HELP files can be found by typing HELPDOC at the TRSDOS command level.


The *HELP Package* consists of five programs. They are:

HELP/BAS: A program which allows you to create customized HELP programs by combining the object code for HELP with a text file created by *Microtext* or *Scipsit*.

HELP/OBJ: The object file described under HELP/BAS.

HELP/DAT: The text file used with HELP/BAS to create the HELP/CMD file included on the disk. You may use this as a base to which you can add additional help commands.

HELP/CMD: A completed HELP program made by HELP/BAS using text file HELP/DAT. It provides instructions for all TRSDOS commands.

HELPDOC/CMD: A complete instruction manual for the *HELP Package*. 

MAZE SWEEP

TRS-80® version by James Garon, translation contest winner
Original program by David Bohlke

Maze Sweep is an arcade-style game for a TRS-80® Model I or III, requiring 16K.

This program is an adaptation of the ATARI® program, *Maze Search*, published in the August, 1980, issue of *SoftSide*. The computer first creates a maze, and then places 25 targets within the maze. The targets appear as "O's (or as "o's if you have lower case); you are represented by an asterisk (*). Using the four arrow keys, you try to hit as many targets as possible before time runs out. Each target is worth 40 points, for a possible score of 1000. In addition, if you are skillful enough to destroy all 25 targets before time runs out, you will earn a bonus: Whatever time remains on the clock will be multiplied by 10 and added to your score.

Note that time passes at a fairly reasonable rate as long as you keep moving. If you stop to catch your breath, however, the timer will count down like crazy.

To hear the sound effects which accompany this game, connect the cable which normally plugs into the "auxiliary" jack of the cassette recorder to an amplifier. Alternatively, you can use a cassette recorder as an amplifier if you have an external speaker: Leave the "auxiliary" cable connected to the recorder, plug the speaker into the recorder's "earphone" jack, and start it running in "record" mode.

Programming Notes

The first unusual thing an alert reader will notice is in line 20. Yes, simple (non-array) variables can be placed in a DIM statement without causing an error. All such variables are given space in memory at DIM time, rather than the time they are first used on the left side of an equals sign. Line 20 DIMs every simple variable in the program. The

motivation for this was explained in my article, *Shedding Light on a DIM Area* in the December, 1979, issue of *PROG-80*. Once an array variable [M(13) in this case] has been DIMensioned, then whenever a new simple variable (such as I, J, or K) is mentioned in the program, something unpleasant happens: The array is moved to make room for the new variable. If the array is large, relocation can waste a significant amount of time.

In this case time is not a concern, since M(13) and P(200) don't take long to move. The important issue here is that M(13) must not be allowed to move at all. Why? Because this particular array will contain the Machine Language sound-effects subroutine. If array M is moved, then so is the subroutine. When a USR call jumps to where the routine used to be and doesn't find it, the results can be catastrophic. 'Nuff said.

Line 30 contains my high score. Beat it if you can.

Lines 40-60 set up the sound-effects subroutine. The routine, as mentioned, is stored in an integer array. If you use this idea yourself, it is essential that the array be an integer type. Use either a DEFINT statement or the "%" sign. Items in an integer array each take up two bytes of memory; thus, two bytes of machine code can be stored in each element of the array. To determine what single number to place in the first array element, take the first two decimal numbers of your Machine Language routine. Let's say these numbers are 205 and 127. Add the first number to 256 times the second number; in this example, that would be 32717. If the number is less than 32768, then use it as is. If not, subtract 65536 from it.

Continue with this method, combining the third and fourth decimal numbers, the fifth and sixth numbers, and so on, until you finish

the last pair. If your program contains an odd number of bytes, pretend that there is a zero after the last byte. Read these numbers into an array, get the VARPTR of the first (zeroth!) element of the array, and that's the entry point for your USR routine. (See lines 50 and 60.) One advantage of this method is that it does not create strings of garbage in the middle of your program (as POKEing into a string does).

The construction of the maze begins in line 200. From a starting point in the center of the screen, one of four directions is chosen randomly. If the way is clear, a line is drawn in this direction. If not, a new direction is chosen. If all directions are blocked, the computer "backs up" to the most recent unblocked position and continues drawing. When this backing-up process causes the computer to return to the starting point in the center of the screen, the maze is complete.

There are visual and auditory aids to following the maze-building: You will be able to see the computer back up along its path whenever it is blocked, and the whole process is accompanied by a tone. When the tone gets lower, it means the computer is successfully adding to the maze. When the tone gets higher, it means the computer is blocked and therefore backing up. When the computer has backed up all the way to the starting point, the tone will simultaneously climb to its highest pitch.

Finally, a word about line 690: This is a test for a blank, CHR\$(32), or an asterisk, CHR\$(42). If neither is found, the player has hit a target. The unusual thing about this line is the omission of THEN when ELSE is used. A line of the form,

```
IF a ELSE b
```

will execute statement "b" only if statement "a" is false. Otherwise, execution continues with the next line. This is perfectly legal syntax in TRS-80® BASIC.

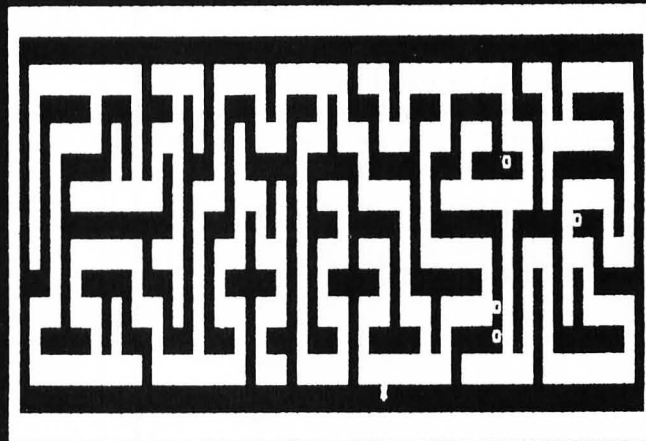
Variables

B, B1, B2: Used to enable/disable BREAK key.
 IS: INKEY\$ variable for movement.
 I, J, K: Loop counters and USR initialization. J and K are used in sound effects; K is also the pointer into the P array.
 M: Byte PEEKed from screen.
 M(n): Array containing Machine Language sound effects subroutine.
 M\$: Graphics block used to build maze.
 N: ASCII value of graphics block (= 191).
 P: Position for PRINTing maze block.
 P(n): Points of maze during construction.
 Q: Offset to next maze block PRINT position.
 SC: Score.
 T: Time remaining.
 V: Video address (= 15360).
 Z: ASCII value of INKEY\$.

Time: 555

Maze Search

Score: 840



```

#####
$      TRS-80 BASIC      $
$      "MAZE SWEEP"     $
$    AUTHOR: DAVID BOHLKE  $
$    TRANG: JAMES GARON   $
$  COPYRIGHT 1982 SOFTSIDE $
#####
    
```

Initialization.

```

10 CLEAR 500:DEFINT A-Z:V=15360:N=191:M#=CHR$(N)
20 DIM B,B1,B2,I,J,K,M,P,Q,SC,SC$,T,Z,M(13),P(200)
    
```

The author's high score. Beat it if you can!

```

30 SC(10)=6640:SC$(10)="James Garon"
    
```

Set up sound routine.

```

40 DATA 32717,19722,15940,26881,-45,
      8237,15613,-11415,11775,
      -736,4160,-20498,-45,201
50 FOR I=0 TO 13:READM(I):NEXT I:VARPTR(M(0)):J=I/256:K=I-J*256
60 IF PEEK(16396)=201 POKE 16526,K:POKE 16527,J ELSE CMD"T":DEFU
SR=I:POKE 14308,0
    
```

Draw the border of the maze and display the time, title, and score.

```

70 CLS:PRINT "Time: 1000"TAB(24)"Maze Search","Score: 0"
80 PRINT "STRING$(63,N);:FOR I=1 TO 13
90 PRINT "CHR$(N)CHR$(253)CHR$(N);:NEXT
100 PRINT "STRING$(62,N);:POKE 16383,N
    
```

Construct the maze.

```

200 P=64*7+31:PRINT@P,M$;:P(0)=P:K=1
210 IF PEEK(V+P+2)-N THEN 260
220 IF PEEK(V+P-2)-N THEN 260
230 IF PEEK(V+P+128)-N THEN 260
240 IF PEEK(V+P-128)-N THEN 260
250 GOTO 350
260 ON RND(4) GOTO 270,280,290,300
270 Q=-1:GOTO 310
280 Q=64:GOTO 310
290 Q=1:GOTO 310
300 Q=-64
310 IF PEEK(V+P+Q*2)=N THEN 260
320 PRINT@P+Q,M$;:PRINT@P+Q*2,M$;:I=USR(1281+K)
330 P=P+Q*2:P(K)=P:K=K+1
340 GOTO 210
350 K=K-1:P=P(K):PRINT@P," ";:I=USR(1281+K):PRINT@P,CHR$(N);
360 IF K=0 THEN 400
370 GOTO 210
    
```

Position the 25 targets (O) and the player (*), after disabling the BREAK key. (In any game using the screen as memory, accidentally pressing the BREAK key is disastrous!)

```

400 GOSUB 9000:FOR I=1 TO 26
410 P=130+RND(11)*64+RND(59):IF PEEK(V+P)-32 THEN 410 ELSE PRINT
@P,"o";:NEXT
    
```

NO ONE ELSE CAN GIVE YOU:

The same high level of Model I/Model III diskette and program compatibility.

True, complete BASIC program chaining with files open and variables saved.

DOSPLUS

DOSPLUS

DOSPLUS

Is the fastest, most powerful, and easiest to operate system on the market. DOSPLUS works! And works right. For the business person and hobbyist, the speed and simplicity cannot be beat. For the BASIC programmer, no one can offer you more than DOSPLUS!

DOSPLUS III 3.3

for the TRS-80® Model III
by Micro-Systems Software, Inc.

Regular \$99.95

SAVE
40%

\$59.88



TRS-80®

Accept arrow-key input from the keyboard and allow the player to move. Decrement the time, and test for hitting a target, running out of time, or hitting the last target.

```
600 T=1000;SC=0;FOR I=1 TO 99:PRINT@P,"*";:PRINT@USR(P),CHR*(143);:NEXT
610 PRINT@P,"*";:I=USR(513+T/4)
620 T=T-1:PRINT@6,T;:IF T=0 THEN 800
630 POKE 16444,0:I#=INKEY#:IF I#="" THEN 620 ELSE Z=ASC(I#):Q=P
640 IF Z=8 THEN Q=P-1
650 IF Z=9 THEN Q=P+1
660 IF Z=10 THEN Q=P+64
670 IF Z=91 THEN Q=P-64
680 M=PEEK(Q+V):IF M=N THEN 610
690 IF M=32 OR M=42 ELSE SC=SC+40:PRINT@55,SC;:PRINT@P," ";:PRINT@Q,"*";:GOSUB 2500
700 PRINT@P," ";:IF SC<1000 THEN P=Q:GOTO 610
```

Game is over. Award any bonus that the player has won.

```
800 PRINT@Q,"*";:T=T#10;SC=SC+T:PRINT@1,"Bonus";:GOSUB 2520
810 FOR I=1 TO 35:PRINT@I," Bonus:"T;:FOR J=1 TO 30:NEXT;NEXT
820 PRINT@I,CHR*(203);:PRINT@55,SC;:GOSUB 2510
```

Maintain and update the top ten players and their scores.

```
2000 M=0;FOR I=1 TO 10
2010 IF SC>SC(I) THEN M=I
2020 NEXT;IF M=0 THEN 2100
2030 PRINT@153,"Congratulations";
2040 PRINT@208,"Your Score is one of the Top Ten.";
2050 PRINT@278,"Please ENTER your Name";
2060 SC#="" :PRINT@479;:INPUT SC#:IF SC#="" THEN 2060
2070 CLS:SC#=LEFT$(SC#,15)
2080 FOR I=1 TO M:SC(I-1)=SC(I):SC#(I-1)=SC#(I):NEXT
2090 SC(M)=SC:SC#(M)=SC#
2100 CLS:PRINT CHR*(23)"Top Ten Scores":PRINT
2110 FOR I=10 TO 1 STEP -1
2120 IF SC#(I)>"" THEN PRINT SC#(I),SC(I) ELSE PRINT"..."," ....
"
2130 NEXT
2140 GOSUB 9010:PRINT:PRINT"Press =ENTER= to play again
2150 IF INKEY#(<)CHR*(13) THEN 2150 ELSE 70
```

Special sound effects.

```
2500 FOR K=0 TO 2:J=USR(4778) OR USR(4742):NEXT:RETURN
2510 FOR K=0 TO 4:J=USR(7764) OR USR(7746):NEXT:RETURN
2520 J=USR(8814):FOR K=0 TO 1:J=USR(8792) OR USR(8778) OR USR(8765) OR USR(8778) OR USR(10328) OR USR(10350):NEXT:RETURN
```

Disable the BREAK key.

```
9000 B=16396:B1=PEEK(B):B2=PEEK(B+1):POKE B,60:POKE B+1,201:RETURN
```

Re-enable the BREAK key.

```
9010 POKE B,B1:POKE B+1,B2:RETURN
```

5

MODIFY EDTASM FOR THE MODEL III

by Randy Hawkins

Second in a two-part series.

There is both good news and bad news about the BASIC program in last month's issue. The good news is that every feature of the editor assembler (*EDTASM*) works as it did before. The only difference you will notice is that the sign-on message is now "MODEL3 EDITOR/ASSEMBLER 1.0" to reflect its new status. The bad news is that *EDTASM* (although it loaded at 1500 baud) will save and load data only at 500 baud. In all my trials operating the program at 1500 baud, the cassette operation was too unreliable at that speed. Too often the object tapes created at 1500 baud would not load correctly with the SYSTEM command and yield a D* or C* error. As a result, the program saves and loads only at 500 baud.

Since I am no expert at Assembly Language programming, I can only speculate as to why this is so. My theory is that the program was written only for 500 baud operation. It could be that the delays between sending a byte out to the cassette port are too time-consuming, so that the timing of the tape is off. The cassette output routine is not called often enough to keep up with the 1500 baud timing rate. What was a reasonable delay between calls at 500 baud is too long at 1500 baud.

Does this mean that you will not be able to use the 1500 baud feature of the Model III for any of your Machine Language programming? Absolutely not! This month we give you an Assembly Language program that can be assembled using the revised editor assembler you completed last month.

The purpose of this month's program is to read a 500 baud SYSTEM tape and rerecord it at 1500 baud. With it, you can do your writing and debugging in 500 baud, and when you have a final, completed program, it can be translated to 1500 baud. By the way, you can also translate any 500 baud SYSTEM tape into the 1500 baud rate using this same program. Now all the programs you wrote for your Model I can be converted to the high-speed Model III rate.

How You Did it

The key to making last month's BASIC program work on the Model III is to replace the portions of the editor assembler that do not work, with routines in the Model III ROM that we know are guaranteed to work. To start with, the keyboard, video and printer routines are located at 43EFH, 4460H and 45AAH respectively. By inserting the appropriate jumps into the ROM at these points, we can insure successful operation of the program. These jumps are to 3024H, 0473H and 03C2H. In addition, in the unused portion that used to be occupied by the editor assembler's printer driver, we can tuck in an initialization routine that loads address 4211H with a zero to set the 500 baud rate, and then jump to the normal starting address, 468AH.

That means the starting address for the revised program will be 45ADH.

The word "MODEL3" is loaded into the title area at 48C9H, and the zero in the version number "1.0" is loaded into address 48E3H. The cursor character (and I have chosen the familiar underline cursor from the Model I, an ASCII 5FH value) is loaded into 430EH.

The subroutine at 4337H in the editor assembler previously turned off the cassette motor. The call to this routine at 46DDH is replaced by a call to ROM routine 01F8H. The call to *EDTASM* routine 43B8H to search for the leader and sync byte has been replaced by a call to 0296H in the ROM. This call is found at 4D57H. The subroutine to blink the asterisks during cassette loads is no longer required since it is done by the ROM routine. Three NOPs are loaded over the call to 4354H located at 4DA6H. The call to routine 43A9H at location 4F34H had the function of outputting to the cassette the leader of 256 zero bytes followed by the A5H sync byte. This is replaced by ROM call 0287H. Finally, all references to subroutines 435DH and 4389H to receive and send a single byte in the "A" register to the cassette have been replaced by a call to 0235H and 0264H, respectively.

After making the proper changes to the editor assembler program, the short program was POKEd in by the BASIC program and saved the revised *EDTASM* to your cassette in the standard SYSTEM format at high cassette speed. The starting address is 4300H and the ending address is 5D00H. As mentioned previously, the revised starting address is now 45ADH.

It would probably be best to make your new copy of the *EDTASM* on a blank cassette rather than over the original version. If something should go wrong you can always repeat the steps in last month's article; once erased, there is no going back with cassette storage.

When you have completed the revision to the editor assembler with last month's BASIC program, you can then proceed with assembling the translator program listed with this article. You could put this general utility program on the back side of the same tape, so that it would always be handy for baud rate conversion. I named the program "SWITCH" to designate its purpose to switch programs from one speed to the other.

Although Radio Shack and other manufacturers have announced plans for improved and enhanced versions of the editor assembler in the near future (or maybe they have already appeared by the time you read this), I was satisfied with the performance of the program as it existed. For the time being, this program will continue to satisfy my needs and save me the expense of a new program with many of the same features. No longer must the Model III owner suffer the indignity of a lack of Assembly Language programming tools — now we have an editor assembler!

TRS-80[®]

```

00100 ;
00110 ; "SWITCH" FOR THE MODEL III TRS-80
00120 ; THIS ROUTINE WILL READ ANY 500 BAUD
00130 ; AT 1500 BAUD.
00140 ;
00150 ; BY RANDY HAWKINS
00160 ;
4400 00170   ORG   4400H
4400 211142 00180 X0  LD   HL,4211H ;SET BAUD RATE
4403 3600   00190   LD   (HL),0   ; TO 500.
4405 218E44 00200   LD   HL,MSG1  ;PRINT STARTING
4408 CDA728 00210   CALL  28A7H  ; MESSAGE.
440B CD6103 00220   CALL  0361H  ;WAIT FOR ENTER.
440E B7     00230   OR    A
440F C2191A 00240   JP    NZ,1A19H ;IF BREAK PRESSED.
4412 CD9602 00250   CALL  0296H  ;READ TAPE LEADER.
4415 216345 00260   LD   HL,START
4418 0607   00270   LD   B,7    ;READ 55H HEADER
441A CD3502 00280 X1  CALL  0235H  ; & THE SIX BYTE
441D 77     00290   LD   (HL),A  ; TITLE.
441E 23     00300   INC  HL
441F 10F9   00310   DJNZ X1
4421 CD3502 00320 X4  CALL  0235H  ;GET NEXT BYTE.
4424 77     00330   LD   (HL),A
4425 23     00340   INC  HL
4426 FE3C   00350   CP    03CH  ;DATA LEADER?
442B 2019   00360   JR    NZ,X3  ;NO - JUMP TO END.
442A CD3502 00370   CALL  0235H
442D 77     00380   LD   (HL),A
442E 23     00390   INC  HL
442F 4F     00400   LD   C,A    ;SAVE NUMBER OF
4430 0603   00410   LD   B,3    ; BYTES TO READ.
4432 CD3502 00420 X8  CALL  0235H  ;READ LOADING
4435 77     00430   LD   (HL),A  ; ADDRESS AND
4436 23     00440   INC  HL    ; CHECKSUM.
4437 10F9   00450   DJNZ X8
4439 41     00460   LD   B,C    ;READ NUMBER OF
443A CD3502 00470 X2  CALL  0235H  ; BYTES IN C REG.
443D 77     00480   LD   (HL),A
443E 23     00490   INC  HL
443F 10F9   00500   DJNZ X2
4441 18DE   00510   JR    X4
4443 CD3502 00520 X3  CALL  0235H  ;LSB OF EXEC ADD.
4446 77     00530   LD   (HL),A
4447 23     00540   INC  HL
4448 CD3502 00550   CALL  0235H  ;MSB OF EXEC ADD.
444B 77     00560   LD   (HL),A
444C E5     00570   PUSH HL
444D CDF801 00580   CALL  01F8H  ;MOTOR OFF.
4450 211142 00590   LD   HL,4211H ;CHANGE BAUD TO
4453 3601   00600   LD   (HL),01H ; HIGH SPEED.
4455 21D444 00610   LD   HL,MSG2
445B CDA728 00620   CALL  28A7H  ;SECOND MESSAGE.
445B CD4900 00630   CALL  0049H  ;WAIT FOR A KEY.
445E D1     00640   PDP  DE
445F CD8702 00650 X7  CALL  0287H  ;WRITE LEADER.
4462 216345 00660   LD   HL,START
4465 7E     00670 X6  LD   A,(HL) ;WRITE TO TAPE THE
4466 CD6402 00680   CALL  0264H  ; BYTES SAVED IN
4469 23     00690   INC  HL    ; MEM STARTING AT

```

```

446A E5     00700   PUSH HL    ; LOCATION START.
446B CD901C 00710   CALL  1C90H ;COMPARE HL & DE.
446E E1     00720   POP  HL
446F 38F4   00730   JR    C,X6  ;NOT DONE YET.
4471 7E     00740   LD   A,(HL)
4472 CD6402 00750   CALL  0264H
4475 23     00760   INC  HL
4476 7E     00770   LD   A,(HL)
4477 CD6402 00780   CALL  0264H
447A CDF801 00790   CALL  01F8H ;MOTOR OFF.
447D D5     00800   PUSH  DE
447E 211945 00810   LD   HL,MSG3 ;LAST MESSAGE.
4481 CDA728 00820   CALL  28A7H
4484 CD6103 00830   CALL  0361H ;WAIT FOR ENTER.
4487 D1     00840   POP  DE
4488 B7     00850   OR    A    ;IF BREAK, THEN
4489 C20044 00860   JP    NZ,X0 ; SAVE IT AGAIN
448C 18D1   00870   JR    X7    ; ELSE START OVER.
448E 1C     00880 MSG1 DEFB  28
448F 1F     00890 DEFB  31
4490 0A     00900 DEFB  10
4491 50     00910 DEFM  'PRESS ENTER WHEN 500'
44A5 20     00920 DEFM  ' BAUD TAPE IS'
44B2 0A     00930 DEFB  10
44B3 52     00940 DEFM  'READY TO PLAY ...'
44C4 20     00950 DEFM  ' BREAK TO QUIT.'
44D3 00     00960 DEFB  0
44D4 0A     00970 MSG2 DEFB  10
44D5 0A     00980 DEFB  10
44D6 50     00990 DEFM  'PREPARE BLANK TAPE'
44E8 20     01000 DEFM  ' TO RECORD THE'
44F6 0A     01010 DEFB  10
44F7 31     01020 DEFM  '1500 BAUD TAPE ...'
4509 20     01030 DEFM  ' PRESS ANY KEY.'
4518 00     01040 DEFB  0
4519 0A     01050 MSG3 DEFB  10
451A 0A     01060 DEFB  10
451B 50     01070 DEFM  'PRESS BREAK TO RESTART'
4531 20     01080 DEFM  ' ROUTINE OR'
453C 0A     01090 DEFB  10
453D 45     01100 DEFM  'ENTER TO MAKE ANOTHER'
4552 20     01110 DEFM  ' 1500 BAUD TAPE.'
4562 00     01120 DEFB  0
4563 00     01130 START DEFB  0
4400 01140   END   X0

```

00000 TOTAL ERRORS

```

MSG1 448E 00880 00200
MSG2 44D4 00970 00610
MSG3 4519 01050 00810
START 4563 01130 00260 00660
X0 4400 00180 00860 01140
X1 441A 00280 00310
X2 443A 00470 00500
X3 4443 00520 00360
X4 4421 00320 00510
X6 4465 00670 00730
X7 445F 00650 00870
X8 4432 00420 00450

```

Parsector V

A review by Marvin Lewis

From Synergistic Solar Inc. System requirements: 16K TRS-80® Model I Level II with cassette. Suggested retail price: tape — \$19.95.

Parsector V is a space war game in which two equally-armed opponents battle for galactic control. Two powerful Motherships must be navigated through the galaxy to conquer individual parsectors by force. The Motherships can launch three types of spacecraft: "fyters," "cruzers," and "bases." The "fyters" patrol their area and attack enemy ships in the vicinity. "Cruzers" jump with powerful engines and capture unoccupied parsectors. "Bases" maintain strong control over captured parsectors. The Motherships can fire high-powered energy beams and short-range weapon spreads. The game ends when one of the Motherships is destroyed or one side has conquered most of the "Galaxy."

So reads the introduction of the game, which was published in 1979.

Parsector V is a fast-moving, graphics space war game played against the computer (three levels of difficulty) or a human opponent. The object of the game is to capture parsectors in a galaxy of five to 25 parsectors.

You start by programming your actions by commands: (1) move your ship, (2) shoot your weapons, (3) launch your ships (of which there are three types), (4) send out a probe, (5) reprint the screen, (6) move one space, (7) shoot one beam, (8) launch one fyter, or (9) weapons spread. You may use any one or several of the commands as long as your time allotment holds out. You and your opponent take turns entering commands and then the execution of those commands takes place as you watch.

The fyters flit around the screen and often engage enemy ships, exploding them. You find yourself

rooting for your ships or chastising them for coming in too close and being destroyed.


Using the supplied galactic chart, you plan your strategies, moving your Mothership from parsector to parsector launching ships and bases. You may move your Mothership in any direction zero to 36 degrees and as many spaces as you want, based on your current energy level.

You can win by owning approximately 68% of the parsectors or by attacking the opponent Mothership and depleting her energy supply. The latter is a dangerous move because you must bring to zero any two of driver energy, weapons' energy, or deflector screens' energy before you can destroy the enemy Mothership.

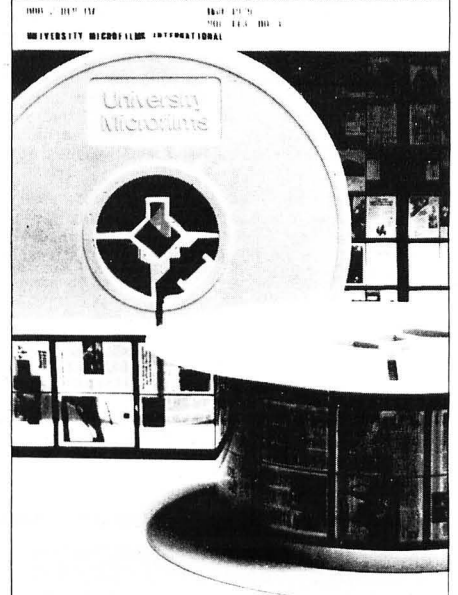
After you own over 50% of the parsectors you receive extra ships when you launch. Now screens fill with exploding fyters and cruzers.

If your Mothership enters a parsector occupied by enemy ships and bases, you have three different weapon functions and you may fire upon those ships, destroying them. Then the ownership of the parsector reverts to you.

The unique split screen allows the players to observe two parsectors, your and your opponent's Motherships, and allows you to watch battles in the two locations. During the "Battle in Progress" phase, the fyters and cruzers engage and the score hit status changes.

Excellent sound effects and good graphics add to a most interesting and challenging game, in either the computer opponent or human opponent (my favorite) situation. There are many tactics that can be used: Do I attack the enemy Mothership as previously described, zip around the galaxy pumping out fyters and cruzers, or secure each parsector with a base and methodically move to the next? Rated a 10, this is indeed an excellent value. 

This publication
is available
in microform.



University Microfilms International

Please send additional information for _____

Name _____

Institution _____

Street _____

City _____

State _____ Zip _____

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

30-32 Mortimer Street
Dept. P.R.
London W1N 7RA
England

SoftSide Selections Ordering Information

Please refer to bound-in order form card for ordering.

USA Orders

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

C.O.D.

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

Canada/Mexico Orders

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

Other Foreign Orders

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

Guarantee

All software is guaranteed to load and run. If you experience difficulties with the product within 30 days, the tape or disk may be returned. Call (603) 673-5288 or 673-0586 for a Return Authorization Number. Any returns without a Return Authorization Number clearly marked on the outside WILL BE REFUSED. Send your properly protected disk or tape to the attention of Customer Service Representative with a note including your name and address.

Liability

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

Prices

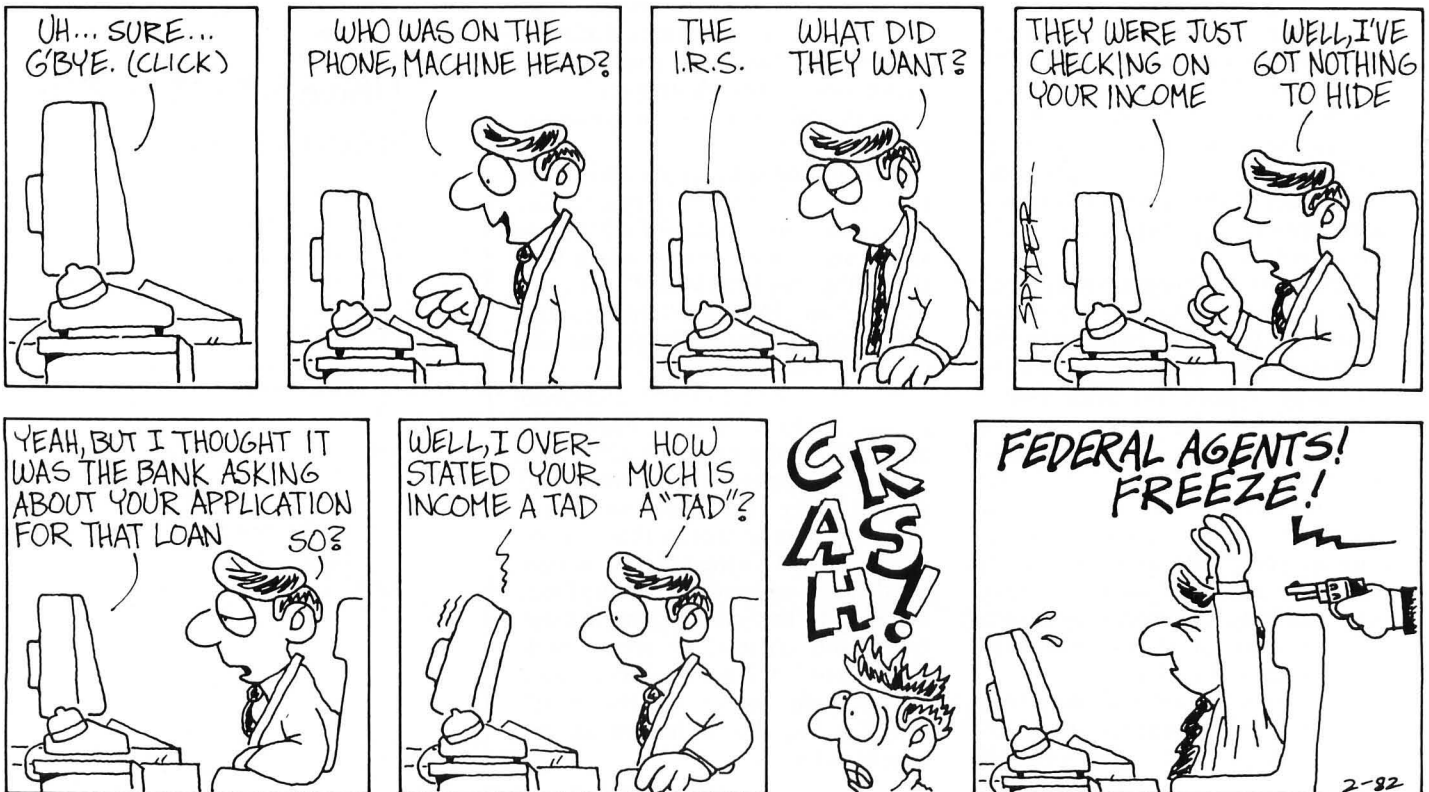
Prices are subject to change without notice. We are not responsible for typographical errors.

Advertisers' Index

Adventure International..... Cover IV
Axlon 68
Continental Adventures 32
ICOM..... 70
National Computer Show..... 14
Small Business Concepts 5
SoftSide Selections 1, 12, 13, 30, 36, 37
..... 70, 72, 76, 79, Cover III
Spectrum Computers 33
Strategic Simulations ...:..... Cover II

MACHINE HEAD

BY SPYDER



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.



The ELIMINATOR

FEATURING
SPECTACULAR

**SUPER
SHARP**

**GRAPHICS
& SOUND**

"THE BEST ARCADE TYPE GAME
I'VE SEEN . . . GREAT FIRE
POWER!"

ROB McCONNELLI,
ARCADER

"NOTHING MATCHES ELIMINATOR
FOR SHEER FUN. I CAN'T STOP
PLAYING!"

PAT HENDERSON,
ARCADER

You're the pilot of The
Eliminator, a space fighter
of the Defender Class. It has
lightning fast response and is
armed to the teeth with
awesome firepower.

But you're outnumbered! And
your attackers are keen. Your
only options are victory or a
grave in space.

 **Adventure** © 1981
INTERNATIONAL

A DIVISION OF SCOTT ADAMS, INC.
BOX 3435, LONGWOOD FL 32750
(305) 862-6917 (QUESTIONS)

ORDER FROM YOUR FAVORITE DEALER

or CALL TOLL FREE (800) 327-7172 (ORDERS ONLY PLEASE)
SHIPPING & HANDLING ARE EXTRA, PRICES SUBJECT TO CHANGE WITHOUT NOTICE
WRITE FOR OUR FREE 150 PROGRAM CATALOG

APPLE VERSION BY JOHN ANDERSON

TRS-80 VERSION BY WAYNE WESTMORELAND & TERRY GILMAN

APPLE 2 - 48K DISK (DOS 3.3 REQ'D.)

042-0134 \$29.95

TRS-80 - 16K TAPE MODEL 1 OR 3

010-0134 \$19.95

TRS-80 - 32K DISK MODEL 1 OR 3

012-0134 \$24.95

ART © 1981 · DON DIXON