SoftSide™

January 1982 Vol. 5, No. 4 THREE DOLLARS

Off and running with GAMBLER

and at the turn it's Vici, Apple Capture, Piazza Hotel...

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!

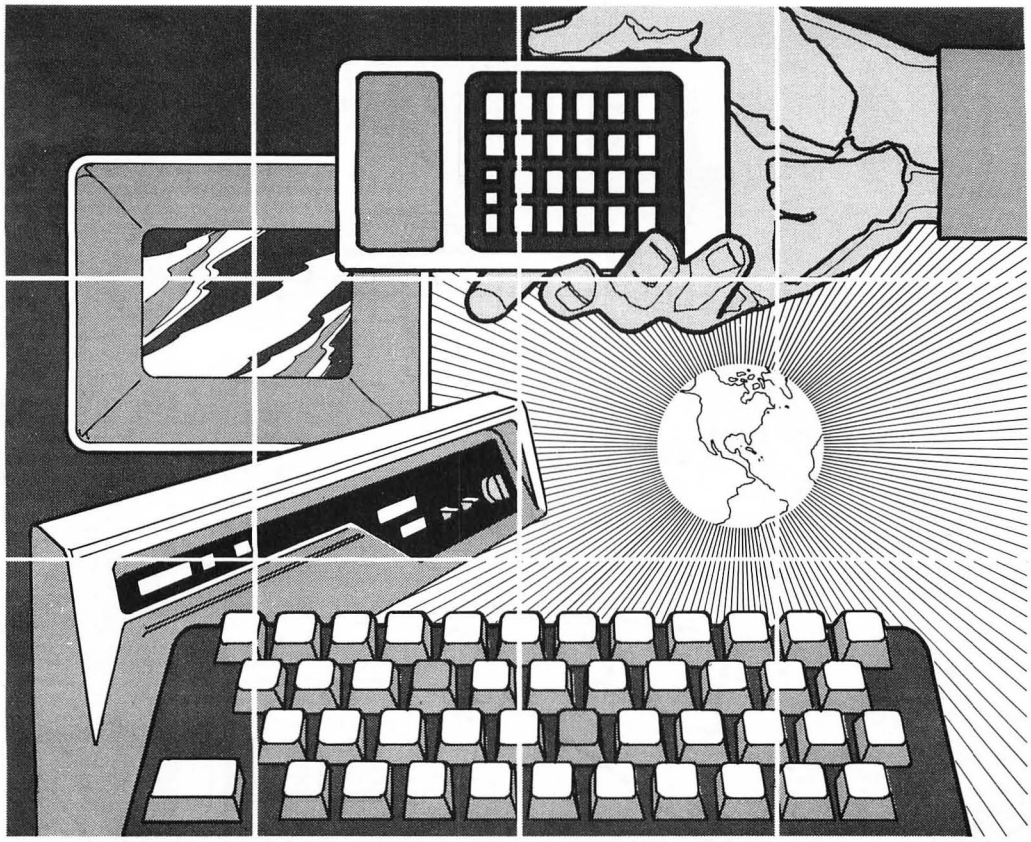
CV gives you the programs offered for your system each month in **SoftSide** on a tape, plus the magazine itself — 12 magazines and 12 tapes per year for just $75.

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.



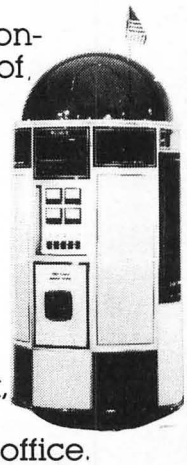
PACIFIC COMPUTER EXPO

ONE PLACE TO SEE IT ALL!

Mini and micro computers, software, graphics, data and word processing, telecommunications, peripheral equipment, supplies and computer services — you'll find it all at the Pacific Computer Expo, January 21 - 23 at the San Diego Convention and Performing Arts Center.

You'll be able to sit down with a computer to evaluate its ability to meet your needs whether business, educational or for your home.

Hands-on workshops will be conducted throughout the three days of this expo and are sponsored by **Radio Shack, Apple Computers, Inc.,** and **IBM.**



Attend seminars designed to advise you where computer technology is heading and its influence on your life today and tomorrow.

And you'll enjoy yourself with computer music, Reggie the Robot, computer art and a host of other electronic gadgetry for home and office.

Plus see a full size display of THE SHUTTLE ORBITER COCKPIT courtesy of **Rockwell International.**

SEE, LEARN, PARTICIPATE AND ENJOY!

PACIFIC COMPUTER EXPO

San Diego Convention
and Performing Arts Center
202 C Street

THURSDAY, JAN. 21, 10 AM - 7 PM
FRIDAY, JAN. 22, 10 AM - 7 PM
SATURDAY, JAN. 23, 10 AM - 7 PM

Adult admission $5.00. Children under 12 free when accompanied by an adult.

PACIFIC COMPUTER EXPO

SAVE $1.00

Present this coupon at the Convention Center box office and save $1.00 on one adult admission.

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 _____

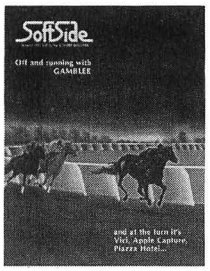
MC# and Interbank#/VISA# _____

Signature _____

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

TRS-80 is a registered trademark of Tandy Corporation

SoftSide™
Volume V — Number Four



Gambler is our featured program this month. Bet on the horse race or take your chances on other games. Do you feel lucky? Illustration by Bill Giese.

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, $30.00/12 issues. USA First Class APO, FPO, Canada, 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, P.O. Box 68, Milford, New Hampshire 03055.

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

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

ARTICLES

REVIEWS

APPLE, ATARI®, TRS-80® PROGRAMS

TRS-80® PROGRAMS

APPLE PROGRAMS

ATARI® PROGRAMS

SOFTSIDE DV

DEPARTMENTS

About This Issue

by Jon Voskuil

❑ It's that time of year when people tend to get sentimental about the past and say hopeful things about the future. But enough is enough, right? Since this is supposed to be about THIS issue, let's get to it.

❑ Top billing at the beginning of this new year goes to DV. For the benefit of the uninitiated, that stands for Disk Version. It also stands for a premium product which contains not only the programs listed in this printed magazine, but also additional high-quality software not available in listed form. We feel that this is the form of the computer magazine of the future, and we're doing what we can to make it a reality today. (Just like the TV ads.)

❑ ATARI® DV subscribers will find *Death Star* to be an intriguing and challenging game, giving them an inside, 3-D view of a maze of passages that must be navigated in order to activate a defective reactor and then escape. *Nuclear Submarine Adventure*, in versions for the Apple and the TRS-80®, is an extensive and creative adventure aboard a modern U.S. submarine, complete with nuclear weapons and all those other goodies that go along with the MAD (Mutual Assured Destruction) philosophy of international politics.

❑ Our three-across cover feature, *Gambler*, is an interesting meld of a variety of games of chance, which may allow you to recoup some of your holiday expenses if you play your cards (and dice, and horses, and lottery tickets) right. It's all done for computer money, of course, and not the real thing. On the other hand, you could dial up the bank on your modem . . .

❑ Also for all three systems is the next installment of the *Microtext* word processor. With the printout module added to last month's original program, you now have the capability of outputting formatted text to a printer for hard copy. Look for other features and enhancements in future issues.

❑ For TRS-80® users, *Vici* and *TRS-Man* should provide lots of entertainment for long winter evenings. *Vici* is enough to make you think you might be able to succeed as President/King/Czar/Shah the next time croup season comes around. And *TRS-Man* is enough to make your computer think it's an arcade game, with all the requisite creatures running around the screen trying to destroy one another.

❑ Apple polishers won't be able to resist *Apple Capture*, a capricious battle of worms in which the goal is to strip the Hi-Res orchard of its 15 scattered fruits. And in a more intellectual vein, the Apple version of *Word Wars* is featured as this month's translation contest winner. You'll find yourself dusting off your dictionary after a few rounds of this one.

❑ ATARI® addicts will have their detective skills tested as they search *Piazza Hotel* for the bomb that may destroy them along with the rest of the guests. And as if this race against time weren't enough, *Number Race* will wear out your joystick as you try to keep numerous characters from getting out of hand on the screen. (If you think that sounds vague, then YOU try describing the crazy game once you've played it for awhile.)

❑ If you're looking for a quickie to type in during a spare half hour, there's also a K-Byter just for you. *Letter Spitter* for the Apple, *Letter Writer* for the ATARI®, and *Row Switch* for the TRS-80®, each has its own unique flavor and is well worth the typing.

❑ In addition to this very full slate of programs (don't expect this every month!), a variety of articles, columns, and other features awaits you. Programming hints, enhancements, and techniques are riddled throughout, as well as reviews, information, philosophy, and all those nice, juicy letters (my favorite part).

That should keep you busy until next month.

REWARD! TRANSLATION APPEAL

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

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

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

Editorial

by Randal L. Kottwitz

The controversy continues. Day after day my desk is flooded with press releases of new products in the microcomputer field. Every manufacturer is trying to one-up the next and when it can't be done, they do their best to emulate the success of the other. "Now with CPM!", "6809, The Chip of the Future!", "The Sixteen Bit Processor Will Change Your Life!" ad infinitum — one must step back and contemplate what would happen if all of the copywriters of these releases were to be brought together in one room. Their claims are certainly less than compatible and yet all believe their way to be the only way.

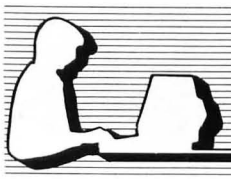
It is out of this sea of confusion that we on the ***SoftSide*** staff must distill our conclusions and take dead aim at the future. There is nothing so constant as change and we believe we can best be prepared for it by anticipating and steering its effect on us and you, our readers.

On the Wednesday before Thanksgiving, the ***SoftSide*** staff gathered and chose the fourth system to be included in upcoming issues of the magazine. Initiating with the June, 1982, issue, ***SoftSide*** will include software and associated material pertinent to the TRS-80® Color Computer. Based on the 6809 CPU, this system has been the subject of a large volume of mail and has shown a phenomenal growth in sales.

What does this mean to you? The most important point is that we **will not** be decreasing the current volume of material for the Apple™, ATARI® and the TRS-80® Models I and III. We **will** be including the TRS-80® Color Computer in our translations and introducing original material for the system as it is available. We, as yet, have not made definite decisions concerning which BASIC (Color and/or Extended) will be the standard or the release dates for Color Computer editions of our media versions. We will keep you posted as we resolve those matters.

At this point we are asking your help. If you are writing software for the Color Computer or have friends involved with the system, please pass the word and submit material as soon as possible. We must have several months' material under preparation simultaneously and would like to assure that we will be able to offer the highest quality product available with the June initialization. Don't hesitate to send for our free Author's Guide and read our "Attention Authors" column carefully.

A brief peek into the future — We currently project the additional system for ***SoftSide*** in 1983 to be the IBM® Personal Computer.



Input

ATARI® INITIATION

Editor's Note: The ATARI® microcomputer is bursting on the scene with skyrocketing numbers. Robert L. Riggs offers the following comments on "Starting Cold with an ATARI® Computer."

People are beginning to discover the amazing capabilities of the ATARI® computers. Very soon, there are going to be thousands of brand-new ATARI® owners, many of whom will be operating their first computer. They'll be looking at a new micro, a new language, new problems and hopefully, new triumphs. This is intended to help smooth the way, somewhat, for the new ATARI® owner. I remember what it was like and I hope that my experiences and some words of advice can help more recent beginners.

How do you get started with this micro marvel? First, read everything that comes with the machine. Set it all up and go through the ATARI® BASIC tutorial, page by page. Run all the little programs and fill in all the blanks. And don't skip the quizzes at the end of the chapters! Maybe you've forgotten how the learning process works with people of *all* ages (not just kids), but I haven't.

Next, study the *BASIC Reference Manual.* After you've read it a couple of times, get a highlighter or two and mark it up for quick access. Go back to the computer store and buy every 6502-based magazine you can find. Check the tables of contents for ATARI® articles and programs. Read them from cover to cover, then study the articles on ATARI®. Type in all the programs and, when you don't understand something, get out the manual and work on it. Don't be afraid to call the Atari Customer Service number (800-672-1430) and yell "Help!"

If you bought the *Assembler-Editor* cartridge, don't even open the box until you've called Atari and requested their "errata" booklet containing all the corrections and don't worry about Assembly Language anyway, until you've mastered the IOCB numbers, GET, PUT, and more. Join an ATARI® Users' Group if there's one near you, and don't be afraid to ask questions.

If you're the super organized type, you can start a card file of articles in the magazines you've collected. Naturally, many of those articles are beyond your present capabilities, but as your expertise increases you're going to need to return to some of them for ideas and concepts that you failed to grasp earlier. You might begin with *Compute!, Creative Computing, Micro,* and ***SoftSide*** magazines. File headings might include: Programs, Game; Programs, Utility; Programming Techniques; Languages; Hardware/Firmware; and Miscellaneous. You'll find that it's far easier when a problem arises to shuffle through your card file than it is to haul out 50 magazines and reread them.

Atari is constantly coming out with new manuals and updates. Give them a call once in a while, and find out what's new. Then order the materials you can use.

Programs that you type in, *especially* when you make an honest attempt to understand what you're typing, are better than those you buy on cassette or disk for the purposes of your computer education. But the IRIDIS series from the Code Works in Goleta, CA, with their booklet of program listings, explanations, and even articles, is a fantastic exception to that rule.

Finally, don't give up! You've purchased a marvelously complex and exciting machine that will amaze and frustrate you as you attempt to solve its secrets, but will also reward you with mind-expanding experiences beyond all your expectations if you keep trying.

Robert L. Riggs
Ventura, CA

POKE MY WHAT?

Dear ***SoftSide***,

I am writing in response to your request for program conversions in the September issue of ***SoftSide*** magazine. I myself consider it a challenge to translate the various Apple and TRS-80® programs into ATARI® BASIC, and have been quite successful with several.

There is one major obstacle that I face in performing these translations. I do not know what ceratin POKEs (and PEEKs) in the Apple and TRS-80® do (cursor on/off, screen clear, etc.). If you could identify their function in the operation of the program, then those of us who would like to translate the more difficult programs could figure out how to do particular POKEs on our respective computers.

As the programs get more complex, (as they inevitably will), these POKEs (and PEEKs) become more important to a successful translation. If this feature were incorporated into a program listing, I'd bet you would be deluged with clever translations.

Also a short feature on graphics comparisons of the three computers would help in translating and reconstructing graphics displays.

As for all the mindless arguing that goes on about which computer is "better" than the other, I would like to add my piece. Each of the computers have their best features. If people spent more time sharing what they do know about their own machines than what they don't know about the others, we might better understand how they all operate.

Eric Stouffer

TAPE AND DISK DUPLICATION

Dear ***SoftSide***,

Thank you so much for finally letting us know what problems are afoot in your software department (ref. Randal Kottwitz column, November issue). I have been a faithful reader of ***SoftSide*** since its first issue, and have always been patient with the then occasional cassette that refused to load. But you have to admit when every Adventure of the Month cassette (for example) has been unusable, a greater problem is present. I have tried all the usual remedies, even investing in a fancy tape digitizer, which I can't recommend highly enough to my friends. Any tape that can't load with the Accu-Data just ain't gonna load, period!

Two ideas come to mind — Using your present "technology," the duplication must be slowed, and a digitizer (analog to digital conversion) must be inserted between master and dupe. Otherwise it is impossible to drive a bank of cassette dupers directly from the computer. A computer-generated program is by far the best available, and you don't even need a fancy certified cassette to boot!

It is indeed unfortunate that the technology does not exist for high-speed digital duplication, but if you examine the audio industry, you'll find any high-speed duplication is unsatisfactory. Hence the move by Mobile Fidelity Sound and others to a real-time duplication of audio material. Most sound engineers will admit that any other method leaves too much room for error.

Perhaps the computer software industry should take a closer look at its objectives. Should they produce a thousand questionable units a day, or 400 high-quality units that the customer can rely on? In the long run, it costs more in replacement and good will than the profit the additional few units can generate.

I understand that some technology just needs time to catch up with the rest of the industry. But the time has come for us to smooth out the wrinkles in these smaller areas like duplication and marketing, which can only lead to better confidence and increased sales.

Thanks for listening.

Scott Daniels
Rowley, MA

Editor's Reply: We appreciate your suggestions. We did some cost analysis on your proposed duplication methods — the resulting figures indicated we could offer a verified, computer-generated, or real-time duplicated cassette subscription for approximately the same price as a DV subscription. A good measure of the increased cost of such methods is the premium price charged for the recordings produced by Mobile Fidelity Sound, and their equivalents in the audio recording field.

ROSES AND THORNS

Dear ***SoftSide***,

I would like to make a few comments about ***SoftSide***. I find that I enjoy the Programming Hints, the articles on technique, and the one liners the most. I also like the K-Byters idea. Since the apparent demise of the Heavy Stuff (last seen about February), I like the short, easy-to-type-in items — I've entered too many games that turned out to be not worth the effort. The best article recently, however, was the one on reverse video in the August issue. It was well-written, a simple project, and the results are phenomenal! I doubt that you can match this article, but keep trying.

I would like to express my disappointment over the trend to disks. I've spent about a grand on the piglet so far ("pig" is a colloquialism for mainframe); I can't find justifying doubling that just to load *Space Invaders* faster (although if I could stay out of the arcade...) I realize that I may be outnumbered.

Although I do read some of the articles on the

ATARI® software, I generally pay no attention to the non-TRS-80® stuff. Especially business articles! My computer is for fun only.

Geoff Dunn
Lenox, MA

Dear *SoftSide*,

My concern deals with the October, 1981, issue of *SoftSide*. In particular, the Apple program entitled, *Super Dairy Farming*. Since May, when this program first appeared in the TRS-80® version, I have been waiting for the Apple translation so that I might type it in to use with my Middle School students. When, in August, the ATARI® translation appeared, I knew that the Apple version of this excellent simulation could not be far behind. When my October issue arrived, and I saw that *Super Dairy Farming* was included, I rushed to my machine, anxious to start typing, only to find that this superb program was only included on the Apple Disk Version of *SoftSide*.

That's what I call "dirty pool." I simply cannot afford to subscribe to the disk version of your magazine (much as I would like to), and I feel that *SoftSide* is depriving me, and my students, of this program.

As much as I enjoy your magazine and eagerly await each issue, I certainly hope that *SoftSide* does not make it a habit of "whetting ones appetite," and then not coming up with the product.

Karen Ann Golubic
Edinboro, PA

Editor's Reply: We're always happy to hear from people who wait eagerly for *SoftSide*, and are sorry to hear of your disappointment concerning the Apple version of *Dairy Farming*. Although we often do publish translations of programs for all three computers, it has never been our intention to do so with every program. The October issue contained listings of three excellent Apple programs apart from *Super Dairy Farming*, which we feel is quite a good value for the $3 cover price (and even better for the $2 subscription price). Our Disk Version is intended to be a premium product, but not one available exclusively to DV subscribers. Back issues of the disks are always available and can be ordered using the card elsewhere in this magazine.

Dear *SoftSide*,

Q: Why is it that each time I open up the new issue of *SoftSide*, I feel like the neglected stepchild?

A: Perhaps it's because I own an ATARI®; and even though I pay the same subcription rate as the owner of a TRS-80® (see, I even refrained from calling it that "other" name), I get the feeling that your organization showers most of its attention and affection on that computer as opposed to mine.

The latest issue of *SoftSide* is a supreme example of this insidious favoritism. I thought for a while that the programs in the magazine were starting to even out a bit — after nearly a year of pampering my computer's older brothers, TRS-80® and Apple — and then, BOOM! I receive the October issue — the long-awaited birthday issue filled with all kinds of exciting things, including an astonishing "breakthrough" in computer software. Unfortunately, the "breakthrough" is apparently meant only for the TRS-80®, with a follow up (of course) for the Apple, but nary a scrap for the poor, neglected younger brother.

Is ATARI® a pariah or something? Why can it not share in this tremendous "breakthrough" which is going to revolutionize the software industry? Doesn't seem like much of a revolution when it ignores a young, but brilliant, member of the family.

But wait — there *was* something for the ATARI® in the October issue. Let's see now, what was it? Oh, yes; an article that explains how an ampersand can be made to look like a square block. It was useful information, to be sure, but rather elementary. Why not show us how to turn those little letters ("g" for ghost, "z" for zombie, etc.) into real, *moving* figures of ghosts, zombies, and skeletons the way the many-talented ATARI® can do so well? There was also a program about a naval battle that tries its darndest to look like a TRS-80® clone (e.g., no sound, no color, no graphics). This, when the rest of the magazine is filled with information about a revolution in the industry that leaves us out!

C'mon guys, your programmers are certainly more clever than that! I realize that ATARI® is just a young member of the family and not nearly as big and important (financially speaking) as his elder brothers, but oh my, what a new-comer! His abilities are new and under-used, but go far beyond those other two. Why not use those abilities to help achieve a *real* revolution in software concepts? At the very least, give us our few scraps of ATARI® programming enhanced with the "ATARI® touch." Don't try to make him imitate his older brother, TRS-80®. You might ruin his creativity.

Above all, don't kick him in some forgotten corner and ignore him. Remember what happened to his well-heeled cousin, Cinderella; or Joseph, of the Coat of Many Colors fame.

I must admit that *I-String* and **Envyrn** look very exciting. On the other hand, it's not going to be much fun watching the other systems take advantage of its possibilities while we ATARI® owners get one more program that produces yet another spectacular, multi-colored, rotating "sunburst."

Bob deWitt

Editor's Reply: As you've stated, the ATARI® computer is a relatively young machine. It has taken it a full year to develop its representation in *SoftSide* with an eye for quality. I think you'll agree we've given it an escalating amount of attention as the quality software is available. As for Envyrn for the ATARI®, I quote the proverbial software developers, "The project is in progress!"

ONE-LINER SUBMISSIONS

Dear *SoftSide*,

I am writing on the subject of one-liners. I feel there is a problem with sending a tape or disk in with a program that only takes 1K (or less) of memory.

I created a one-liner; I would like to type it, but the rules require me to send a tape or disk with the one-liner on it. I believe you could get more people to send in one-liners if you would allow them to send it in on a typewritten sheet of paper. I believe that sending a typed letter with a one-liner in it is better economically because some people cannot afford to buy a new tape (or disk) just for a one-liner.

I would appreciate a reply on this subject. Thank you.

Robert Little
Los Alamos, NM

Editor's Reply: Obviously, any program written on media can be reviewed more easily, especially the lengthy ones. However, we have never required that a one-liner submission be on tape or disk.

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.

A joyous 1982! This year promises to be yet another "Gateway to the future." We hear that statement every year, and every year it becomes a little bit truer. The acceleration of technology in our culture seems to be never ending and we, as computerists, continue to be on the leading edge of the tidal wave.

We would like to extend a belated welcome to the newest member of the ***SoftSide*** staff, Lynn Wood. Lynn joined us last month as our new production manager and illustrator. We're sure her contribution to ***SoftSide*** will delight your eyes.

As I've stated before, the readers' input to ***SoftSide*** is extremely important. ***SoftSide*** is truly unique in the manner in which it speaks to you, our readers, as well as your computers. In order to continue to speak with an accurate understanding of our audience, we must continually update our files concerning your interests, lifestyles and equipment. Next month, we will be asking you to respond to the most important poll we've ever printed. There are many possible changes being considered for ***SoftSide***. These include such possibilities as a sectionalized, system specific magazine which would contain more pertinent material for each system than we currently print. Another possibility would be to print the actual line listings, with the exception of one top quality program translated for all systems, in a separate code book to be published concurrently with each issue of the magazine. By isolating the line listings in a third "media edition," if you will, we could actually issue more programs per system per month, printing only the introductory articles, documentation, features, programming articles, and other related material in the "slick" magazine. I must emphasize that these are not definite plans, only projected possibilities. However, before any of these matters may be brought to intelligent conclusions, we must know how you make use of the current ***SoftSide***. Watch for the poll next month and take a few moments to fill it out and return it to us. A magazine can remain vital only as long as it speaks to the needs of its readers. Please help us maintain our vitality and we'll do our best to continue to please both you and your computer.

I have recently attended several seminars concerning electronic/database publishing. There is no doubt that electronic data communication is truly the "voice of tomorrow." However, currently, there are many technological barriers such as slow baud rates, incompatibility of data formats, etc., hindering the growth rate of such communications systems. In an effort to prepare you and us for maximum utilization of this new form of publishing, ***SoftSide*** will soon initiate a series of columns to take the reader from ground zero to a greater understanding of electronic data communications. I am anxious to hear your opinions and desired results from such a series. In addition, we are in the process of compiling databases for future use on ***SoftSide*** DV and for dial-up access. We are currently researching the best methods for compilation of such data and will keep you informed as more details are available.

Some of you who have been with us since December of 1978 may recognize the photo/illustration on page 59 of this issue, the opening of *Vici*. The same photo was used as the cover of that December issue, featuring a program called *Santa Paravia En Fiumaccio. Santa Paravia* has since become a successful commercial program for Instant Software, translated for five different computers. The principles of the simulation live on. There are a large number of schools using the simulation as an introduction to economics. Many advances have been made in BASIC programming since we published *Santa Paravia* and we believe *Vici* to be an excellent example of utilization of those advances.

Take a seat at your computer and enjoy this month's jam-packed issue. Happy Hacking!

Randal L. Kottwitz

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

StarFighter

BY SPARKY STARKS

The PENULTIMATE Space War Game is HERE!

TRS-80*
Model 1 & 3
16K and up — Tape
010-0102 . . . . . $24.95
32K and up — Disk
012-0102 . . . . . $29.95

Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.

BOX 3435 • LONGWOOD, FL 32750 • (305) 862-6917

ORDER FROM YOUR FAVORITE DEALER
or CALL **TOLL FREE (800) 327-7172**

SHIPPING & HANDLING CHARGES ARE EXTRA
PRICES SUBJECT TO CHANGE WITHOUT NOTICE

*TRS-80 is registered Trademark of the Tandy Corp.

Hints & Enhancements

ONE BYTE, MORE OR LESS

I was very interested in Bob Howell's *The One Liner Challenge* (*SoftSide* 9/81) and grateful for his suggestion of saving the program as a REM to preserve the abbreviations, etc. Although the one idea I've tried just wouldn't fit, my style requires frequent packing of many statements in one line and I can use the idea at those times.

Working in this field is like a curve approaching an asymptote; no matter how strongly you feel you've squeezed the last excess byte out, someone always comes along and shows you how to do better. For starters, Bob's POKE82,0: gives you two extra characters per line for a total of six. However it costs nine, for a net loss of three. There's no need to give up those six, though. The POKE, like the REM, is needed only for editing, not at RUN time, and needn't be in the program at all; it can be typed first in direct mode.

A tighter method of alternating a variable between 0 and 1 than A = ABS(A-1) is A = 1-A (and it's faster too).

The statement that ATARI® BASIC will handle FORI = 1TON*2STEP2 because ". . . the last character before STEP is numeric" is inaccurate. It's because the item *begins* with a numeric symbol. In this case the first character is also the last, but it's the *first* that's important.

For performing the first part of the program twice, F.I = 1TO2...N.I takes three less spaces and can handle more go-rounds than the ON A GOTO even with the improved alternator.

When it comes down to it and you're still a few characters short, you can drop the REM and even the line number if necessary (I don't know how you could get it on tape or disk then, but it doesn't seem to violate any rule.)

I'd thought of using the "." abbreviation for REM, but LISTing changed it back. This brings up another point though. All the micro mags seem to be unable to adequately represent a space in typeset copy. Even when giving examples of coding with and without a space, it almost takes a magnifying glass to see the difference. Can't you instruct your typesetters to use two or three spaces if necessary to approximate the width of an average character?

I've always wondered why you restricted your One Liners to display demos, and was glad to see that requirement dropped for your new K-Byters. The One Liner attempt described above was expanded and probably fits in 1K. It demonstates an ATARI® special feature and is also a useful program. However I don't have any spots on the wall I need to hide.

Next to articles/programs which give me new information on my ATARI® (all too rare since the sample issue that convinced me to subscribe), my favorite part of *SoftSide* is *The Sensuous Programmer* by "J". I admire the breadth of his knowledge of the three micros he covers. However in his September column, its depth failed him and made the ATARI® seem decidedly inferior to the others when it comes to keyboard input. Firstly, his suggested six statement routine (as opposed to two or three for the other micros) checks for signs or a decimal point, but the presence of these does not insure that a digit follows, so the program could still bomb. The statement that RETURN-only input leaves the input string unchanged is true about some micros (I remember reading it also) but it isn't the ATARI® .

The answer to the input problem is to use the TRAP keyword, covered in another context later in the column, but rather disparaged. Here's how:

```
100 TRAP 10000:INPUT A
10000 GOTO 100
```

This will handle numeric input directly, and resets the TRAP before each use. One caution: unless the following routine(s) set their own TRAPs, turn off the TRAP after the successful input with a "TRAP n" where "n" is any value between 32767 and 65535; this will prevent surprise returns to 100 if an error occurs elsewhere in the program.

Although it appears not to be part of the problem "J" was addressing, I would like to add that in most circumstances it is advisable to warn the user of a mistake via sound, color, and/or displayed message, so he or she can correct the error.

Bill Friedman

Editor's Reply: We are trying to use LISTings from a printer whenever possible. This reduces the likelihood of input errors. When a LISTing might be typeset, we try to approximate the actual spacing from the CRT as accurately as possible.

FLIP IT AGAIN, ATARI®

In your September, 1981, issue, the game *Flip It* will run on the ATARI® with only 16K memory by eliminating the lines 5000-5130 inclusive.

The following lines should be changed. (This will put the sound through the TV speaker instead of the computer speaker.)

```
1165 FOR Z=1 TO 6:SOUND 3,50,10,10:NEX
T Z:SOUND 3,0,0,0:RETURN
2003 FOR B=1 TO 60:X=A(B):X1=7*(X-INT(
X/10)*10)+7:Y1=INT(X/10)*5-3:SOUND 3,7
5,10,3:SOUND 3,0,0,0:LOCATE X1,Y1,Z
2050 FOR X=1 TO 20:NEXT X:PRINT :PRINT
 "WANT TO PLAY AGAIN?";:POKE 764,255
3000 FOR C=1 TO 8:X1=7*(X-INT(X/10)*10
)+6:Y1=INT(X/10)*5-3:SOUND 3,75,10,1:S
OUND 3,0,0,0:GOSUB 2990+20*C:NEXT C:RE
TURN
```

Walter J. Korzyk
Albany, NY

MORE GAS PER LINE

This One Liner calculates gas mileage. The only real sneaky thing is the use of the "PRINT USING" command, which rounds the answer, as well as limits the number of decimals that print out.

```
888 LPRINT" START"TAB(12)"END"TAB(22);
:LPRINT"GAL"TAB(31)"MPG":INPUT"FIRST F
ILLUP";F:FORI=1TO2STEP0:INPUT"NEXT FIL
LUP";S:INPUT"GALLONS";G:M=S-F:PRINT"MP
G = ";:PRINTUSING"##.###";M/G:LPRINTFT
AB(10)STAB(20)GTAB(30);:LPRINTUSING"##
.###";M/G:F=S:NEXTI
```

Geoff Dunn
Lenox, MA

AT A POKES NOTICE

In reference to your article *Mixed Text and Graphics in HGR2,* (September, 1981), there is a simple solution to relocating your Applesoft program (almost) anywhere you desire.

Locations 67,68 HEX, or POKE 103,104, contain the starting address, low order byte, and high order byte of an Applesoft program. For example, by changing locations 67,68 (normally 01,08 HEX) to 01,60 your program will be loaded just above Hi-Res page two.

One problem is that these changes can't be made from within your original program. Where there's a will, there's a way. Simple, just write a short program to load your program and make these changes. Here's a short example:

```
5  INPUT "NAME OF PROGRAM ";NM$
10  POKE 103,1: POKE 104,96
20  PRINT  CHR$ (4);"RUN ";NM$
```

From then on, until you reboot or power down, any Applesoft program will be loaded at the new address. Save it to disk and it will be ready at a POKE's notice.

A few other locations to POKE around with, should the desire strike you:

HEX	DEC	Explanation
67,68	103,104	Start of program
69,6A	105,106	Start of variables
6B,6C	107,108	Start of arrays
6D,6E	109,110	End of arrays
AF,B0	175,176	End of program
	Integer	
CA,CB	58,59	Start of program
4C,4D	76,77	End of program
	Binary	
AA72,AA73		Start of program
AA60,AA61		Length of program

I hope this helps. You guys put out one heck of a magazine. Have fun.

W. K. "Rocky" Shryock
Grants Pass, OR

ATARI® JOYSTICK AND THE TRS-80®

I have recently purchased an Alpha Products ATARI® joystick that has been adapted to work

on any TRS-80® Model I. The joystick works quite well, especially with the numerous arcade games by Big Five Software, all of which I own.

I do have a modification of the October, 1981, ***SoftSide*** program, *ABM Command* by Arnold E. van Beverhoudt, Jr., allowing the game to utilize the TRS-80® ATARI® joystick. Lines 6020 to 6110 need be changed to the following:

```
6020 W=INP(0)
6030 IF W=239 GOTO 6150
6040 IF W=254 GOTO 6170
6050 IF W=247 GOTO 6190
6060 IF W=253 GOTO 6210
6070 IF W=251 GOTO 6230
6080 IF W=250 GOTO 6250
6090 IF W=246 GOTO 6280
6100 IF W=249 GOTO 6310
6110 IF W=245 GOTO 6340
```

If you own a TRS-80® Model I with Expansion Interface, and if you use cassette port #2 for sound, change line 3060 so that you POKE 14308,1.

The *ABM Command* game is slow, and I ran my *Packer* program from Cottage Software, which speeds up the game considerably.

John M. Delaney, Jr.
Wood River, IL

IMPROVING WORD CHALLENGE

I recently entered the TRS-80® version of *Word Challenge* from the September, 1981, issue of ***SoftSide***. It is a very enjoyable and challenging game, however, there were two areas I felt could be improved.

The first is the flashing word definition and graphics display when a game is won. This is nice to watch the first two or three times, but after that I found that it tended to slow the game down. I made the following changes to correct this.

ADD — 214 IFINKEY$ = CHR$(13)THEN220
DELETE — 230
CHANGE — 235 GOTO8

These changes allow the player to break out of the game won display by pressing the enter key.

I also had difficulty remembering which player's turn it was. The following modified line 79 prints a left-pointing arrow beside the initial of the player whose turn it is.

```
79 IFPL=0THENPRINT@898,CHR$(93);:PRINT
@902," ";ELSEPRINT@902,CHR$(93);:PRINT
@898," ";:T=T+1:IFT)=215THENPL=PL+1:PR
INT@X+Y*64," ";:X=4:Y=1:D=1:GOTO78
```

Alan Haines
Victoria, BC, Canada

BRRRRRACKKK

Although I have found very little about the ATARI® which I don't like, one of the minor irritations is the repeated "brrrrrackkk" sound it makes while LOADing or SAVEing to and from cassette. I have found a way to overcome this.

Once the machine is CSAVEing or CLOADing, you can press CTRL and the number 2 (at the same time), which is designed to give you the buzzer sound. You can then turn the sound down on the television so you won't have to listen to the repeated data stream doing its data-streaming, and when the READY shows up on the screen, the buzzer command will be executed. This is your machine, notifying you that your CSAVE or CLOAD operation has just been completed.

I hope this can be of use to the many ATARI® users who read your magazine. They will probably find, as I have, that it is more calming to the family's nerves than to the user's.

Leonard E. Buchanan
APO, NY

A BEAST BY ANY OTHER NAME . . .

Since my first encounter with ***SoftSide*** in June of 1979, I have admired and enjoyed your magazine. At first I did not totally agree that the "new" format was such a great idea, but now I have begun to think that the change was for the better.

In *Hints and Enhancements* in the November issue, I stumbled on New Dungeon for *Quest I* and immediately typed it in. I found there was a lot more to expect from the game, other than it was great. I started, went into a room, and had to combat five Wraiths and the treasure was only "worthless odds and ends." I also found that the words Orc, Giant, etc. were getting old. If it is any help or enhancement to your readers, here are the changes that I made.

You could edit all of the data statements, deleting all of the last elements and delete the ",T1(X)" or just change line 615 to read:

```
615 READ M1(X),M2(X),T1(X):T1(X)=RND(8
):NEXTX
```

This will generate a random treasure.

I also changed line 110 to read:

```
110 DATA GIANT RAT,R,2,GOBLIN,G,3,HOBG
OBLIN,H,4,MINOTAUR,M,6,BUGBEAR,B,7,TRO
LL,T,8,VAMPIRE,V,9,YOUNG DRAGON,D,9.9
```

I'd like to thank you at ***SoftSide*** for putting out a great magazine, with few advertisements. Keep up the good work! And if I may, thanks to Brian Reynolds and Tigre Wenrich.

Kevin McHugh

APPLE PUFF AND TAB

I enjoyed Randi J. Rost's Apple Puff in *Hints and Enhancements* (***SoftSide***, November, 1981). However Randi's program will only run error-free if A$ in line 20 is exactly 40 characters long. Any message length less than 40 characters will cause a BAD SUBSCRIPT error in line 50.

I made the following changes to allow any message from 1 to 40 characters in length to run error free.
ADD line 45 L = LEN(A$)
CHANGE line 50 from B$ = A$(2,40) to B$ = A$(2,L)
CHANGE line 60 from B$(40) = A$(1,1) to B$(L) = A$(1,1)

If the message is going to be less than 40 characters in length, add an appropriate TAB statement after the VTAB statement in line 30. This will center the message on the screen, giving a nice, neat-appearing message. Line 20 could be changed to an INPUT statement (with appropriate screen formatting commands) to allow new messages to be used by merely rerunning the program.

One correction (I realize it was a typo): line 80 should read GOTO 30, not 60 TO 30.

As a subscriber, I thoroughly approve the changing format. You're good, and getting better. Thanks for an interesting magazine.

Nick DiMarco
Gillette, NJ

RENUMBER YOUR APPLE

The *RENUMBER* utility program supplied with Apple DOS 3.2 and 3.3 has an insidious bug — after you use *RENUMBER*, your program may still appear to run perfectly, so you may not even notice that your program's operations have been altered! *RENUMBER* will correctly change all line number references to agree with the new line numbers. Unfortunately, *RENUMBER* may also alter any number in an arithmetic expression which (a) follows an asterisk (the multiply operator), and (b) has the same value as a pre-RENUMBER line number.

I obtained the corrections for the DOS 3.2 version from the Apple Hotline in May of 1980. I just discovered that the problem still exists in the DOS 3.3 version, and I am still seeing letters in various magazines from perplexed Apple users.

The fixes for the DOS 3.2 and DOS 3.3 versions of the program are similar — they involve swapping two data values in the program, as follows:

	DOS 3.2	DOS 3.3
From BASIC	POKE 4815, 172 : POKE 4816,171	POKE 4789, 172 : POKE 4790, 171
or		
From monitor	* 12CF : AC AB	* 12B5: AC AB

To permanently correct the *RENUMBER* program, you must:
(a) LOAD *RENUMBER*
(b) Do the two POKEs for your version of DOS
(c) SAVE *RENUMBER*

All Apple owners should take note of these fixes — even if you don't use *RENUMBER*, you'll be able to help out the next guy, who may not have read about this problem! For your future reference, Apple dealers have a loose-leaf notebook which answers commonly-asked questions — including, "What's wrong with RENUMBER?" You just have to know to ask.

The latest issue of *Apple Orchard* indicates that the two locations to be POKEd for RAM Applesoft *RENUMBER* are 14342 and 14343.

Robert C. Leedom
Glenwood, MD

SAVE FORMAT

If you have ever recorded programs on a cassette and have forgotten what format they were saved in (i.e. BASIC, machine code, or EDTASM source code) or have forgotten the filename, then this short program is for you.

```
10 CLEAR 600
20 INPUT$-1,A$
30 IF ASC(MID$(A$,2,1))=211 PRINT"BASI
C PROGRAM CODE - FILENAME: ";MID$(A$,4
,1):END
40 IF ASC(A$)=85 PRINT"SYSTEM - MACHIN
E CODE - FILENAME: ";MID$(A$,2,6):END
50 IF ASC(A$)=211 PRINT"EDTASM SOURCE
CODE - FILENAME: ";MID$(A$,2,6):END
60 PRINT"FORMAT UNKNOWN - DATA?"
70 END
```

Quentin Barnes
Chester, IL

Your Adventures

July Adventure of the Month
Alien Adventure

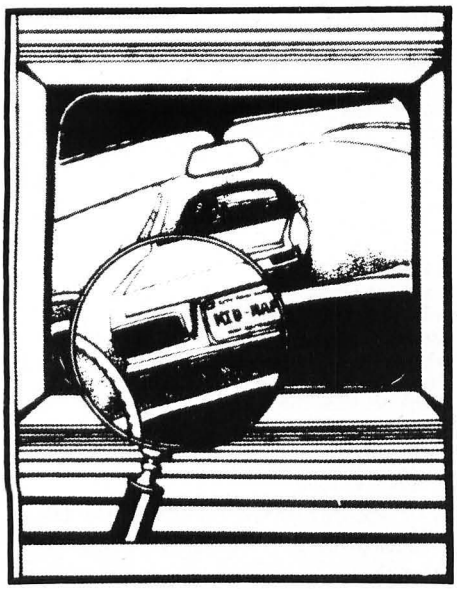
You are the sole survivor of a crew on a mission to deliver a cargo of oil to Earth. A crash landing has left you stranded on a small planet, harshly alien but rich in lead, gold and platinum. You must find provisions and a means of leaving the planet. But beware of the THING that massacred your crew!

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.

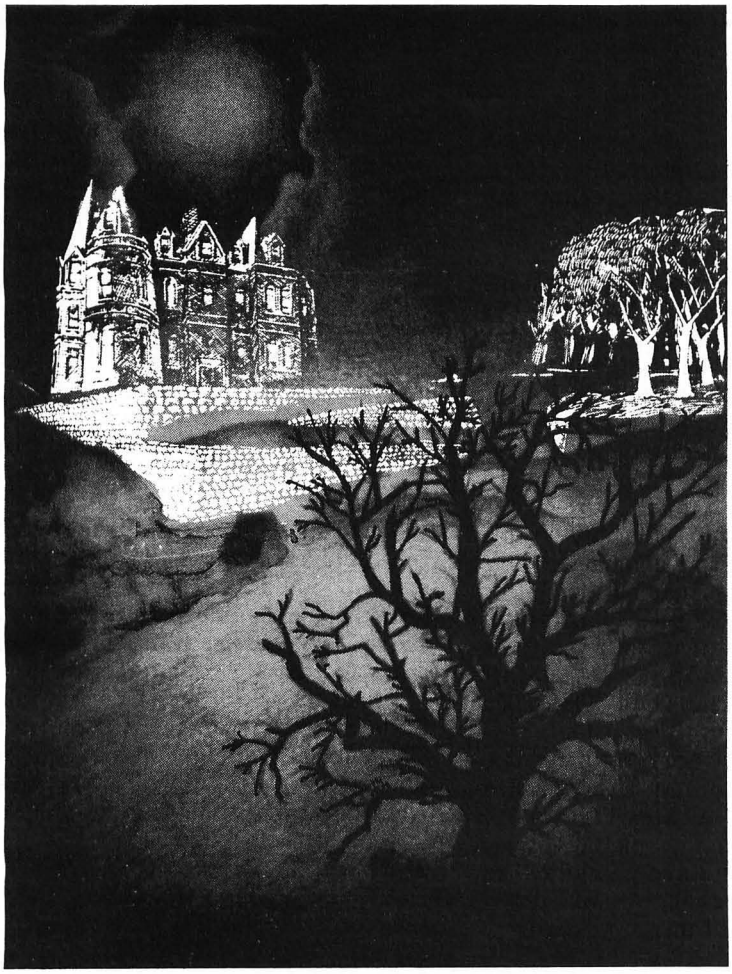
November Adventure of the Month
Around the World in Eighty Days Adventure

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.

will start here —



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.

JANUARY ADVENTURE OF THE MONTH: WINDSLOE MANSION ADVENTURE

In the dungeon of Windsloe Mansion the world-famous Pumpkin Man is being held prisoner. An underground passage connects the mansion to the Blair house, whose owners have agreed to assist you in rescuing the prisoner. Will you succeed in overcoming both the human and the supernatural creatures who are rumored to inhabit Windsloe Mansion?

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)**

_____
_____
_____

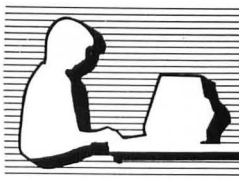
- ☐ Cassette ($6)
- ☐ Disk ($9)

■ **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.



Say Yoho

by Scott Adams

When we last met, I was talking about the difficulties I surmounted while transferring my Adventures from the Apple to the ATARI®. I left off with the Apple source and data files residing on an eight inch TRS-80® Model II disk.

The problem, as you remember, with transferring from the Apple directly to the ATARI®, was that the speed on the Apple RS232 card was constant. However, the ATARI® would get data into its off-board RS232 port, transfer it over a special audio serial line to the main computer, into my tranfer program, then into memory. My transfer program would then output the data through the audio serial port to the disk drive. The disk drive would then transfer the data to the diskette.

The ATARI® is a really fine personal computer, but because of its use of an audio serial bus to link up to outside peripherals, it has the slowest disk drives in the industry. For example, a full disk copy from one drive to another takes about eight minutes to complete! What I needed was a program which would transmit some data and wait before continuing.

Miracle of miracles there exists such a program for the Model II TRS-80®. It's *ST-80 III* by Lance Micklus. Lance's industry standard program has a special mode called "Veriprompt™" which will output one line of data, then wait for the receiving program to transmit a character before continuing.

It was a simple matter to write an ATARI® BASIC program to receive a line of data, store it on disk, and then request more to be sent. In this manner, I went ahead and transmitted all the Assembly Language source files and different Adventure data files to the ATARI®. Finally, phase one was done and I was ready to start re-writing Adventures to run on the ATARI® computer.

Except for simple things such as page heading controls and so on, I found the assembler source code from the Apple version of my Adventures would assemble nicely using the standard ATARI® *Editor/Assembler* cartridge. So I was ready to start getting the program to run on the ATARI®.

Since both the Apple and ATARI® use the 6502 computer processing chip, the only real changes I needed to make were to the I/O (input/output), as each computer handled them differently. The major I/O sections that my Adventures use are: save game, player input, and screen output.

Save game was simple to convert using the routines built into the ATARI®'s operating system and was no problem at all. The same was true for the keyboard input routine.

When I got to the screen routines, it started getting interesting. Adventures use split screens on the Apple and I wanted to maintain this feature on the ATARI®. Unfortunately, the Apple has a memory map video area. This means a certain area of regular memory is set aside as a window. If you place a character into this memory, it then automatically appears on the CRT screen. The ATARI® on the other hand doesn't set aside a particular area of memory for its video output. Instead it has a pointer which can be moved to any area in memory, and depending on the graphics mode selected and even on the number of carriage returns per screen, will determine the amount of memory used!

Every time (in graphics mode 0 which is just standard text) a carriage return is put to the screen, a logical line is set up. Each logical line may be up to 160 characters long though total screen size is only 40 characters by 24 lines (physical screen size is same as the Apple).

What I finally ended up doing was to allow the ATARI® to build me a standard 40 x 24 character display list and give me the starting address. Thereafter I accessed that memory area directly like I had done in the Apple and that was that!

The final thing I did on the ATARI® was define an alternate character set. The ATARI® has a pointer which directs the hardware to a memory area that contains the bit mapped font for the character set. It is simple to define one's own character set in RAM and point the pointer there. This I did with my Adventure script font I had developed and it worked quite nicely! So I finally had my Adventure series running on the ATARI®. I'm tempted to tell you of my latest conversion project which was to the new Commodore VIC-20 computer, but I think I will spare you!

Next, I'd like to share something nice that just happened here at Adventure International. I'm sure by now you realize software publishing is a big business and with the rewards come many of the headaches.

Licensing, lawsuits, and copyrights are some of the myriad problems that come up. Another is names! You come up with a great new program and it's ready to market. Now you must pick a name for it. The problem that arises is: Has someone else already used that name?

There is really no fool-proof way to find out. You can run searches which start at $800 a name and up, you can look through source books, and ask at shows. But basically it just comes down to crossing your fingers!

Recently, we introduced with much fanfare a fantastic game for the TRS-80® called *Starfighter*. We've spent thousands of dollars advertising, packaging and promoting the program in general. Well, yesterday, I got a bit of a shock.

A letter in my "IN" basket contained two sheets. The first was a copy of page 163 in *80 Microcomputing*, July, 1980. In the middle was an ad and circled on the ad was *STAR FIGHTERS — Destroy Robot Fighter's Mother Ship.

Oh great, I thought, we've got a problem now. Just what we need! I then turned to the accompanying letter and read:

Mr. Scott Adams:

We probably should not both have "Star Fighter" games on the market (see enclosed magazine page).

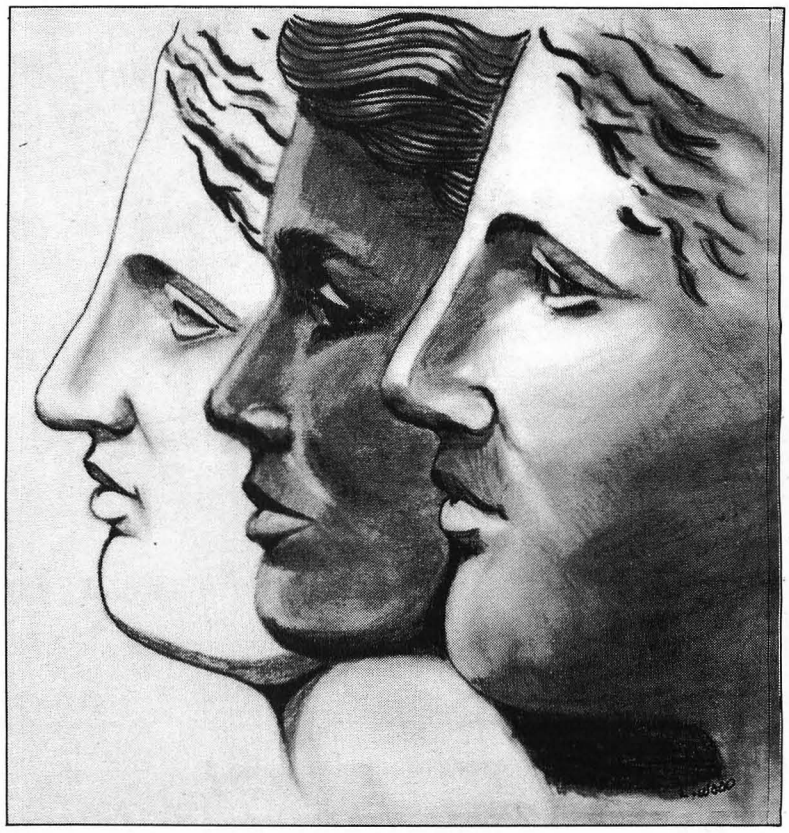
Since I'm a nice guy, I will take steps to change the name of my program in future ads and documentation. If you're a nice guy, you could be of assistance to me in the marketing and distribution of games for the TRS-80® Color Computer. Enclosed is a cassette and instructions for one such game, "Brickaway."

Thanks
Britt Monk

All I can say is — Thanks Britt. You are indeed a nice guy! 'Til next month . . . Happy Adventuring!

And Then There Were Nine

by "J"



Bugs. In the natural order, they far outnumber us more intelligent creatures. And despite their miniscule size, it requires a phenomenal amount of our time and other resources just to maintain an uneasy standoff with them.

So it is with the other kind of bug, the kind that every computer user knows intimately. Just when you think you've got them beat, they attack with a vengeance. Often in the most embarrassingly painful places.

Most programmers spend a lot of their programming time (and some of their sleeping time) poking around in their programs (and those of other programmers) trying to unearth bugs, worms, and other undesirables. People who edit computer software magazines (not to mention any particular ones) cringe inside every time a program listing is published, knowing that any program over six bytes long is bound to have some kind of flaw that survives detection by programmer, reviewer, tester, and copy editor. Add to those flaws the ones introduced in the glare of a bare bulb at 3:00 a.m. via the bloodshot eyes, numbed brain, and stiff fingers of a haggard typist, and you have the stuff of an exterminator's wildest fantasy.

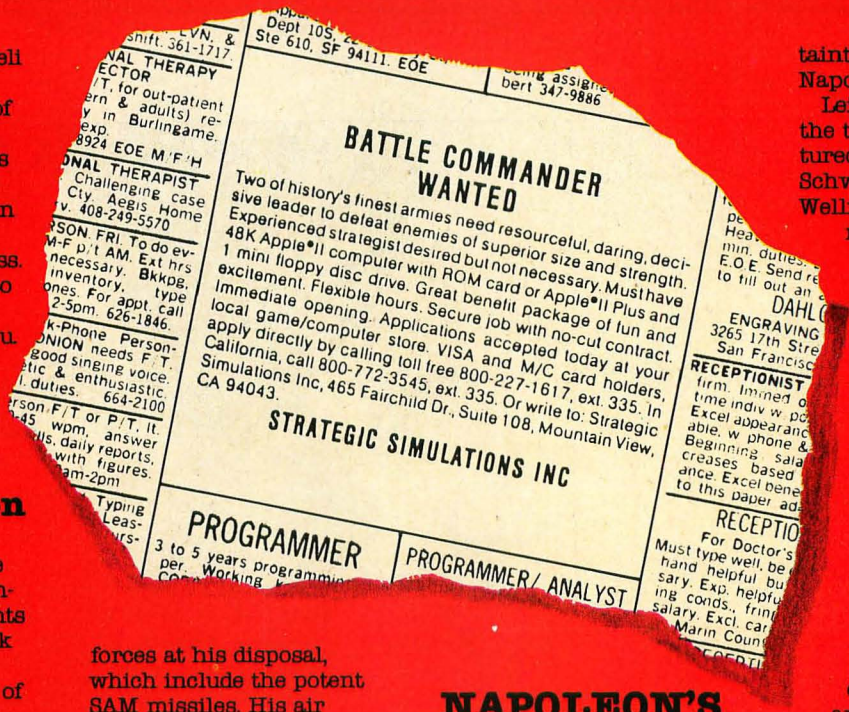
There are at least four distinct types of bugs which attack computer programs. One type is produced by a flaw in the programmer's logic, so that under some circumstances the program does not in fact do what it is supposed to do. A second type is caused by mistyped characters of one sort or another — which can have a wide variety of effects, from the very obvious to the very subtle. A third type is caused by trying to use the program with a system or in an application which was not intended by the programmer. And a fourth type is caused by some flaw or glitch in the magnetic and electronic media where the program is being stored and processed. Although these last two are not really bugs in the program itself, the user still perceives them as bugs, since the program does not run properly.

Now, if bugs would only identify themselves as being one of these three types, we debuggers would lose a lot less sleep. Unfortunately, when a problem surfaces in a program, it's often far from obvious where the source of the problem lies. Even an apparently "obvious" system error message such as "SYNTAX ERROR IN 1000" might result not from a mistyped character, but from such things as a failing or loose RAM or ROM chip, a power line surge or nearby static discharge, a glitch in the tape or disk from which the program was loaded, the alteration or addition of some other program line, or trying to run the program in a system with inadequate memory.

Debugging is an exercise in detective work. The most obvious clues, such as the error messages generated by the computer, are sometimes misleading (though usually very helpful). And often the majority of your clues are quite circumstantial in nature, leaving you to do a great deal of guessing and groping for the real culprit. My only

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 uncertainties inherent in Napoleonic warfare.

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

SSI

STRATEGIC SIMULATIONS INC

Sensuous Programmer

continued from page 15

background in detective work comes from watching reruns of *Columbo* and *The Rockford Files*, so I won't carry this analogy much further. But it seems to me that debugging involves a lot of hunches, educated guesses, and dumb luck, as well as routine gathering of evidence, and that one's debugging skills improve greatly with experience.

Assuming that you encounter some kind of problem with a program, what should you do? As far as I know, there is no comprehensive, foolproof system for ferreting out bugs. I offer the following suggestions as creative alternatives to the other, more destructive options such as angry letters, blind panic, and kicking the family dog.

1. If you are typing in a program from a printed listing, assume from the very beginning that you'll probably end up spending as much time finding typographical errors as you spent typing it in. Most people who believe that they are exceptions to this rule, aren't.

This suggestion highlights the importance of a person's attitude while trying to get a program to run properly. In most cases, published programs do work as advertised; and if the program you've typed in from the listing crashes within a minute or two, there's an awfully good chance that the problem was introduced by the human interface between the listing and the keyboard. (And we both know who that is.) Sure, there are errors in published listings. Sometimes whole lines or sections of a listing are accidentally cropped out, and occasionally there are horrible errors in logic or syntax. Once you've been burned by such a program, it's easy to get paranoid about bugs. But having a positive, problem-solving attitude, rather than a negative, critical one is the first step in successful debugging. Think of what you can learn by discovering the mistakes that other people make, as well as your own!

2. If the program you're troubleshooting has a lot of DATA lines, that's a good place to start looking for trouble. A very common technique in programs is the use of FOR-NEXT loops to READ a quantity of data. The data might then be stored in one or more arrays, or perhaps POKEd directly into certain memory locations to create a Machine Language subroutine, shape table, or the like. If a single character in one of these DATA lines is mistyped, the results can be quite dramatically disastrous — or extremely subtle and difficult to detect.

Consider some examples. A missing comma, or a look-alike period or semicolon substituted for a comma, will cause two successive items of data to be read as one. If the data consist of numbers to be POKEd into memory, this will often generate an ILLEGAL QUANTITY error in the line which is trying to do the POKEing (not in the DATA line). This happens because only integers from 0 through 255 can be POKEd into a single memory location. If the numbers 102 and 66 are supposed to be READ and POKEd, but a missing comma causes them to be read together as the number 10266, or if a period makes them into the number 102.66, then you're in trouble. A semicolon between them would cause the computer to see them as a string quantity rather than a number, and probably cause a TYPE MISMATCH error — this time, in the line which is READing the data.

If the program you're trouble-shooting has a lot of DATA lines, that's a good place to start looking for trouble.

Of course, even if there is no error detected by the computer as the erroneous datum is being read, some kind of error will result later. It might be an OUT OF DATA message from the computer (again, in the READing line), as it tries to read past the end of the data. Or it might be a system lockup or other strange behavior resulting from the wrong values having been POKEd into memory or assigned to variables.

Other common errors in DATA lines include extra commas; confusion between characters such as O and 0, I and 1, B and 8, and S and 5; and items (or whole screen lines) which get either repeated or omitted. Some such mistypings may generate system error messages which help to pinpoint the problem, but often the errors will go undetected until they cause strange behavior later in the program. Some mistypings are even invisible in a listing: a shifted character may list as a normal one, but have a different ASCII value; and an inadvertantly typed control character will not list at all. It's worth it to retype a suspicious line (even all the DATA lines) before beginning your long walk off that short pier.

Often closely related to problems in DATA lines are errors associated with statements such as PEEK, POKE, CALL, USR, VARPTR, and ADR. As mentioned above, POKEing wrong values into a Machine Language subroutine can wreak all sorts of havoc. Likewise, using any of these statements with erroneous arguments will surely cause weird and wonderful (or terrible) things to happen. If the keyboard locks up, or you start seeing garbage displayed on the screen, or your disks start rebooting, or you get nonsensical error messages, look closely at lines which use these keywords.

3. Be aware that the line in which the computer says an error occurred, is not necessarily the line that has a problem. (In some cases, the specified line might not even exist!) Error messages such as "BAD SUBSCRIPT", "ILLEGAL QUANTITY", "OVERFLOW", and "OUT OF MEMORY" often occur in lines which have no errors in them. What the computer is usually telling you is that the value of some variable used in that line is not what it should be. Which means that you have to backtrack through the program to find out where that variable was given an improper value.

In such cases, the first task is to determine what value in the specified line is illegal. For example, consider this line:

```
1000 POKE X, VAL( MID$
( R$(J),N-2, 3))
```

There are all sorts of possibilities here for illegal values. X must be within the memory range of your computer (expressed as either a positive or negative number); J must be non-negative, and within the DIMensioned size of R$; the quantity N-2 must be at least 1, and no greater than the length of R$(J) minus 2; and the VALue of the three specified characters in R$(J) must be a non-negative integer less than 256.

If an error message is generated by this line, and it's evident that there are no mistyped characters in it, then the first step is to find the values of all the variables and expressions in it. Since the variables all retain their values after the error message is generated (unless you alter a program line or type a com-

mand such as CLEAR or RUN), all you need to do is tell the computer to PRINT the values you need. Just type in

```
PRINT X, J, N-2, LEN(R$(J)),
MID$(R$(J),N-2,3)
```

and press RETURN/ENTER, and those five values will be displayed on the screen. Or at least three of them. The fourth or fifth one might provoke the same error message that you just saw when running the program.

Once you've determined what value or values are illegal, you can then start the detective work, tracing the flow of the program back to those lines where the values of those variables were assigned or changed. Of course, if the values were computed using other variables, then those other variables have to be traced back as well. This untangling continues until you've located a line where there is some kind of error. Sounds like fun, no? If you can't find the bad line, it's sometimes helpful to put in temporary STOP or END or PRINT statements at various places in the program, to check on the value of some variable at those points. Using the TRACE or TRON command before RUNning the program can also be of great help in following the flow of the program prior to the error.

4. Maybe this is obvious, but I'll mention it anyway. When your computer is gracious enough to give you an error message, look up that message in your reference manuals and read about the various things that can provoke it. You may be surprised to find out how ambiguous some system error messages are.

Apple's "OUT OF MEMORY" message, for example, can be generated by any of several conditions. The most obvious one is that there's not enough free memory for the program or for the variables. But the most obvious is not necessarily the most common. Other causes include improper use of FOR-NEXT loops, GOSUBs, and parentheses. Any of these, if "nested" more than a given number of levels deep, will exhaust the limited memory assigned to keeping track of such things (the system "stacks"). So this error message could be generated, for example, by a mistyped variable name in a FOR or NEXT statement, or by the wrong number of parentheses in a mathematical expression. Read the manuals to make sure you're covering all the possibilities.

5. The confusion that can exist among characters which look very much alike has already been mentioned. It's worth reiterating, specifically with regard to variable names. If you're trying to find one mistyped character in ten or twenty thousand, it's really easy to miss an E1 that's supposed to be EI, or a B8 that's supposed to be a BB. Such miniscule differences can cause aggravations unlimited.

6. Making minor modifications to a listed program is a common thing to do: Combining short lines, reformatting PRINT statements, renumbering lines, adding REMarks as you go along. Usually such things have no ill effects, if you know more or less what you're doing. But even a modification as simple as adding a REM can cause a program to crash. This can happen if the programmer has written the program to occupy only a certain portion of the computer's memory, and swelling it by just a few bytes could invade another part of memory reserved for another use. If the program POKEs in a Machine Language routine, or uses a graphics screen, which overwrites part of the program itself, obviously something is going to blow up sooner or later. This phenomenon is usually detectable by LISTing the program and finding part of it missing or filled with garbage at the end. The solution, at least, is relatively simple: Type it in the way it was listed in the first place!

But even a modification as simple as adding a REM can cause a program to crash.

7. Some errors are really incomprehensible. Of course, all errors probably seem incomprehensible to the uninitiated; but some just plain don't make sense. A specific example will illustrate the point.

The October issue of ***SoftSide*** contains the program *Leyte*. When I load this program into my Apple from the ***SoftSide*** disk, it runs fine. But when I SAVE it to tape, turn the computer off and back on, and LOAD it back in from tape without DOS in memory, it crashes the first time I press a key. The error message says "ILLEGAL QUANTITY ERROR IN 260". Line 260, and the two preceding lines, are as follows:

```
250 GET I$: IF ASC(I$) < 49 OR
ASC(I$) > 52 THEN 270
255 FOR X = 1 TO 1: NEXT X
260 G = VAL(I$): ON G GOTO
280,480,620,730
```

The variable I$ contains the character corresponding to the key I have just pressed — in this case, the digit 1. The variable G, then, should have a corresponding value. When I type "PRINT I$, VAL(I$), G" the computer displays the following values on the screen:

```
1        1        1.11111111E+16
```

Picking myself up off the floor, I ask myself why G should be equal to something over eleven quadrillion, when it has just been set equal to a quantity whose value is one. After retyping the lines several times, reloading the program from tape and disk several more times, and repeating the whole procedure on a different Apple with the same result, I conclude that I am actually in the Twilight Zone and everything will be different when the full moon sets in a few hours.

The next day, I did manage to get the thing to run by first setting HIMEM to some arbitrary value a few bytes below the top of memory. Why? Who knows? Finally, my sanity was rescued by a little blurb in the Beagle Bros. Micro Software *DOS Boss* manual, mentioning this "GET bug" in Applesoft. Although they didn't provide an explanation of the bug ("The explanation for this is really boring"), they did provide a fix. If you insert the statement "I$=I$" after the GET, before using the VAL function, then all is well. Would anyone out there like to provide the reason for this one??

The moral is this: Yes, there really are system bugs that can cause some very strange errors from time to time. (There ought to be an UNFATHOMABLE ERROR IN LINE NNN message to point these out, don't you think?)

Well, if this column hasn't made you into a confirmed entomophobe, nothing will. Next month, I'll be celebrating the tenth installment of *The Sensuous Programmer*. Maybe I'll take the day off instead of writing it. . .

Entertainment Tomorrow

by Allen L. Wold and Fred D'Ignazio

BOARD GAMES UNLIMITED

As computer games continue to improve and develop, will there be any place in the future for traditional board games? We think there will be, though with microcomputers embedded in them, they'll be quite different from what they are today. Let's examine one possibility, incorporating four different ideas.

Our hypothetical game is similar to many present-day games, such as *Squad Leader*. It is a war game, in which each player controls units representing squads, single trucks or tanks, and special wagons. Each player attempts to control parts of a town or countryside as represented by the game board. But in other ways our game is quite different.

For example, when the board is unfolded there is only a simple white surface without markings of any kind. It is a giant Liquid Crystal Display (LCD), no thicker than a traditional war game board.

At the side of the board are two units like pocket calculators, with LCD readouts and a full set of flat, touch-sensitive keys, which can either enter letters or numbers, or designate or perform special functions — not sine or square root, but "Attack," "Move," "Range," "Radar," "Overrun," "Rally," and so on.

The third item is like a long, thin pair of dividers, connected by a wire to the board and keyboards. One leg is red, the other is black, and at the joint between the two legs is an analog device which knows the angle to which the legs are separated.

The pieces which come with the game are blank on one side, and on the other are marked like traditional cardboard war game unit counters, but without any numbers.

Even if all you've ever played is *Monopoly*, then you know about counting spaces, rolling dice, keeping score, and so on. Our hypothetical game turns all that bookkeeping over to a microcomputer embedded in the game, and leaves you with only the playing to worry about.

Our game can plug into the wall or run on a battery. You turn on a switch at the side and the once white board becomes a full-color representation of the playing surface, with buildings, roads, forests, rivers — almost like an aerial photograph, except the artwork makes identification of each object easier, and there are contour lines representing elevations. What's more, the picture-map displayed is different for each variation of the game you want to play.

Next, you and your opponent set out your pieces, plain side up. Neither of you knows what the other's pieces represent, whether tank, squad, leader, machine gun, or dummy. When all the pieces are placed, the board displays a dot by any piece which is in sight of your opponent's pieces, and those you flip face up.

First turn. You move first. But there are no hexes printed on the board, no squares, so how do you know how far to move?

You simply slide each piece across the board, along its intended path. Your side-display tells you when each piece has gone as far as it can go, depending on type, terrain, and current condition. The side-display also tells you whether an opposing piece has spotted you as you moved, giving it a chance to fire at you.

After you have moved all the pieces you want and are able to, those which are capable may fire on enemy targets. Your anti-tank gun on a hill would like to take out that armored vehicle down in the valley, but is it in sight? In range?

You take those long-legged dividers now, and place the black leg on the piece representing your anti-tank gun, and the red leg on the unit counter representing the enemy armored vehicle. Then you touch the "Attack" key on your keyboard, and the side-display tells you the results. No measuring, no dice rolling, no looking up tables and forgetting which weapon is modified how by which target under which circumstances. The computer knows all and does it for you.

The board displays a red ring around all units which were destroyed. The pieces are removed from play after all attacks are resolved, and the rings disappear. Units which were merely damaged are similarly marked by an orange ring, and demoralized units by a yellow ring. The board displays these rings wherever the pieces might be, and will continue to display them until their condition changes.

After you finish your attack you may attempt to repair or rally any units which were damaged or demoralized on the last turn. You touch each unit with both legs of the divider together, and touch the appropriate key on your keyboard. If the attempt works, the rings disappear or change color. If not, they remain the same.

During your opponent's turn he or she slides a piece representing a tank across the board. It goes through woods, crosses a short clear stretch, then goes into the shelter of a building.

But while it was in the clear, one of your units had it in sight and you have the option to fire. You use the dividers as you did when it was your turn, but the computer embedded in the game calculates not only the type of attack, the type of target, the armor if any, and the range and elevation modifiers, but also the amount of time the target unit was in sight, and adjusts your chance to hit accordingly.

If your opponent tries to move a piece that has already moved, the computer won't allow it. If you try to fire out of range, the computer scores no hit. If you can't see the unit, you can try radar, using the dividers again.

You refer to no tables. You count no hexes or inches. You roll no dice. You simply command your troops to move and fire, attack and defend, and hope that you understand your opponent's strategy well enough to be the victor. All the mechanics of the game have been taken care of by the computer. All you have to do is play.

In most war games the element which slows play the most is the necessity of referring frequently to tables which tell the players the outcomes of certain actions, according to the throw of dice. There is a constant struggle in game design to try to satisfy two conflicting criteria: The tables must be easy to read and apply, yet

must also incorporate a number of factors bearing on the results of a player's decisions, and therefore be complex. It's the old playability vs. realism conflict — if you achieve one, you lose the other for the most part.

In our example, all tables are incorporated in the game's computer, as is a sophisticated random number generator which replaces the ubiquitous pair of dice.

One of the benefits is that the computer can handle complex tables which take into consideration more factors than a person can normally keep track of. There is more to combat than simply a machine gun firing at a truck at six hexes. Is the truck moving and how fast? What are the elevation differences? Are there bushes or buildings screening it? What type of ammunition is the machine gun using? Is your squad under fire from somewhere else and therefore distracted, or has it sighted in on that section of road and therefore is more accurate? What is the morale of the troops on both sides? How good is the commanding officer? How experienced are the troops? Are they fresh or tired? Have they taken damage earlier in the game? Is this a holding action or a do-or-die situation?

It is impossible to include all those factors in a single printed table, to be cross-indexed with a single die roll. Instead several tables must be consulted, each with a choice of one of several columns as determined by other factors listed in other tables. Then the die roll is modified according to still other tables. Certain conditions on the board have to be supplied from memory, and so on. Mistakes are easy and not uncommon. Misinterpretations are easier. So, for the sake of simplicity, and ease of play, many of the above factors are simply left out, and a less realistic conclusion is reached.

Despite the use of polyhedral dice in role-playing games, most board games and war games still use the old six-siders. But in real life the chance of something happening is not always a nice neat 1/6, or 1/36, or whatever the percent chance is if two dice are rolled and added.

But if you use an electronic dice-rolling generator, you can have 17-sided dice, if you wish. The chances of various outcomes can be given realistic weights, and the true chances rolled for. There is no need to round up or down to the next whole number, as every present-day board game requires. If the odds are 27 to 13, that's what they are, not 2 to 1 or 3 to 1.

The computer will also keep track of "victory points," your score in the game — both those which are awarded permanently, i.e., for destroying an enemy tank; and those which are awarded conditionally, i.e., for holding a hill, which may change in the course of the game. When you've finished your last turn (the computer keeps track of that too) then the score is displayed.

In many games, additional units come on at various times. The computer will remind you of this. It reminds you of night-turns, and other times when different conditions apply. It reminds you that some demoralized troops are now recovered, that you need to reset your long-range artillery, and that attempts at repair can now be made.

The board uses two very different technologies. The first is that instead of cardboard or plastic sheets with a printed picture on the surface, we have a flat LCD board. The map is computer-generated color graphics, which can represent a wide range of terrains and conditions. When a tank is wrecked, the board displays a wreck. When a bridge is built, the board displays a bridge. When an obstacle is removed, the board shows clear ground. The condition of the player's forces and the terrain the board represents, is displayed by the computer without any effort on the player's part, and is always up-to-date.

Status tables are displayed along the side, indicating the player's score, or other information not necessarily kept secret. These are also always up-to-date. And if you change scenarios, that mountain in the middle is simply eliminated. Where there was no city before, a new one springs up.

There is an advantage to a horizontal board over a vertical CRT screen. The CRT screen is small, and everybody sits in front. The LCD board can be quite large, and players sit on various sides. Also, the aesthetics of a large full-color board cannot be denied, and playing pieces tend to fall off a vertical surface.

The second characteristic of the board is that it is also a sensing device. Each of the unit counters has embedded in it a kind of magnetic signature. The computerized board senses the magnetic patterns, and knows what the piece represents. Because the map on the board is created by a program, the computer knows where each piece is, and can relate it to other pieces on the board, as well as to the terrain the board is displaying at the moment.

If you indicate an attack on a piece that is behind a building, the computer tells you can't see it. If you have an elevation advantage, the computer knows it. All the information on the board is contained in a set of arrays within the computer's memory, and while you still should know what your unit counters can see and what they can't, the computer takes everything into consideration without you having to type it in.

The computer can also tell whether a piece is at full strength, demoralized, half-strength, and so on. It remembers which pieces have fired, which have moved, and which have not. You might still misjudge range and firepower or accessibility of target, but those problems, after all, have not yet been solved in real war.

The long-legged dividers are perhaps the most unusual piece of equipment, based on the rulers used in miniatures war gaming. But instead of a typical ruler marked off in inches, centimeters, or any other scale, the analog device where the two legs join measures the angle to which the legs are opened, and therefore the distance between the two ends. The legs are colored distinctively — black and red in our example.

Each leg has a sensor which can read the identity of a piece, or a location on the board. The black leg is keyed to the pieces or locations of the player whose turn it is to move, the red to the opponent's. Thus, when one leg is placed on one piece and the other leg on another, the computer knows which two pieces or locations are referred to, and which belongs to which side. No need to type anything in.

This permits several interesting possibilities — radar for instance. Place the black leg on your radar unit and swing the red leg in an arc. The computer knows what terrain is within the arc of the radar unit, and can spot "targets" for you, though not necessarily telling you what they are.

The dividers would also be used to rally demoralized troops, or give status on ammunition, casualties, and so on, by placing both legs on the piece in question and touching the appropriate function key. All information is displayed on that player's side-display, where the opponent cannot see it. Here is where the computer itself is contained in a few dedicated microchips.

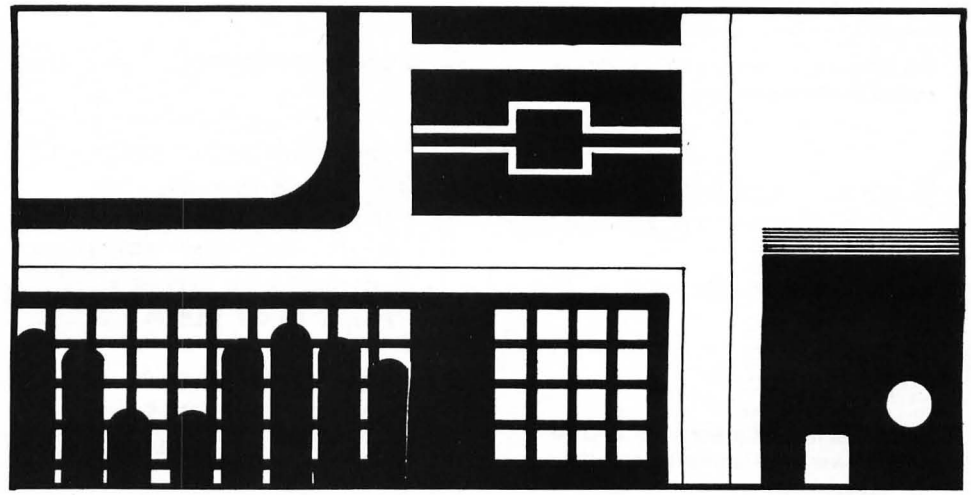
Modern board games, as well-designed as they might be, still take a lot of work to play. With tables, movement-counting, written moves, and record keeping in some cases, more time is spent on the mechanics of the game than on making decisions which determine the outcome of the game. But it won't be that way forever. Our hypothetical game is not that far in the future.

Microtext 1.1

by Jon Voskuil

Program Documentation Textwriter, Part 2: Printout Module

***Microtext 1.1* is a word processor program for a 16K Apple (with Applesoft), ATARI®, or TRS-80® (Model I or III).**



Last month we began the development of a simple BASIC word processor program. That first (and main) installment included the code needed to enter text (with lines automatically broken between words), save and recall text using tape or disk, review the entered text on the screen, and delete unwanted lines to the end of the file.

As outlined last month, ***SoftSide*** now requires that all program documentation be submitted on tape or disk in machine-readable and -storable form. The version of *Microtext* published in that issue has all the capabilities which are absolutely necessary to serve that purpose. But there are many features that can be added, to make the program much more useful as a general-purpose word processor. The most urgently needed of these is obviously a printout routine, to enable you to format and print the text on a printer.

The following printout module allows you to do this. The printout option is selected in the same manner as the other ones are, in this case using a CTRL-P or CLEAR-P. The parameters you can specify for the printout are the left and right margins and the line spacing. The routine assumes that your pages are 66 lines long, and prints six blank lines at the end of a page. This means that you should position the paper down about two lines from the top of the page before starting to print. It would be a simple matter to add provisions to specify the top and bottom margins, but for the sake of simplicity we've omitted that seldom-used option. (Lines 7070 and 7615 would need to be changed, and another INPUT line or two added in the 7010-7050 range.)

An additional feature for the Apple and TRS-80® versions is a routine to convert all-upper-case text to mixed upper and lower case for printout. This is not needed for the ATARI® version, since lower-case text can be entered directly; nor is it needed for a TRS-80® Model III, for the same reason. If you have a Model III, you can make the following changes in the listed program:

line 50: omit ':GOTO 100'
lines 59-74: delete
line 7050: delete
line 7130: change to 'LPRINT TAB(LM);P$;'
line 7140: delete
line 7590: change to 'LPRINT TAB(LM);PP$;'
line 7600: delete

Using the lower-case conversion option, all text will be printed in lower-case characters unless flagged by an '@' symbol. Whenever the routine encounters an '@' in the text, it will capitalize the following character; when it encounters a double '@@', it will capitalize the following word. The shift key is used in the normal way on the TRS-80® Model I to indicate capital letters; you'll notice that when you use it, the letter you type will automatically be displayed on the screen with an '@' preceding it.

On the Apple, the shift key does not function in this way; instead, the ESC key is used for capitalization (a convention used by many Apple word processors). Pressing ESC will display an '@' on the screen, indicating that the next letter typed will be capitalized; pressing it twice will capitalize the whole word. The '@' key itself may also be used, but that's a lot more trouble since it requires pressing two keys (shift-P). You'll be surprised at how quickly you'll get used to using the ESC key in this way.

A weakness of the lower-case conversion routine is its speed. Since it must check for '@' symbols within each line of text, and add 32 to the ASCII value of each unflagged character to convert it to lower case, it does take a moment to convert each line. (On the other hand, it's still a lot faster than typing!) Suggestions from readers on improving the BASIC routine, or substituting a Machine Language subroutine to do the job, are welcome.

The next module for *Microtext* will add a line editing feature which will allow you to delete and change text much more easily than is now possible. The editing module is under development, and will be published in these pages in the near future.

Additional Variables

CC: ASCII of a character.
E: Error code.
LC: Lower-case flag.
LIN: Number of the current line on the page.
LL: Line length.
LM: Left margin.
LOK, LOCK: Shift-lock flag.
LP: Length of P$.
LS: Line spacing.
N0, N1, N32, N64, N91: Used in place of constants for increased speed.
P$, PP$: Text to be printed.
RM: Right margin.
UC: Upper-case flag.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      APPLESOFT BASIC        $
$      'MICROTEXT 1.1'        $
$    AUTHOR: JON VOSKUIL      $
$     (C) 1982  SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

The following are lines to be changed or added to Microtext 1.0:

```
3  REM $     MICROTEXT 1.1     $
20  VTAB 8: PRINT  TAB( 8)"M I C
    R O T E X T   1 . 1"
50  FOR Z = 1 TO 3000: NEXT Z: GOTO
    100
```

Subroutine to print a line of upper-case text in lower case (unless preceded by '@' to capitalize one letter or '@@' to capitalize whole word).

```
59  REM LOWERCASE PRINT RTN
60 X$ = "": IF PP$ = "" THEN 74
62 LOCK = N0:LC = N1: FOR K = N1 TO
     LEN (PP$):CC =  ASC ( MID$
    (PP$,K,N1)): IF CC = N32 THEN
    LOCK = N0
64  IF CC <  > N64 THEN 70
66  IF LC = N0 THEN LOCK = N1
68 LC = N0: GOTO 72
70  PRINT  CHR$ (CC + N32 * (LC AND
    CC > N64 AND CC < N91) * ( NOT
    LOCK));:LC = N1
72  NEXT
74  RETURN
```

A few miscellaneous changes.

```
100  HOME : ONERR  GOTO 20000
140 CHAR = 1:N0 = 0:N1 = 1:N32 =
    32:N64 = 64:N91 = 91
170  PRINT D$;"MON C,I,O": HOME
210  VTAB 24: HTAB 1: INVERSE : PRINT
    " SAVE:^S  REVIEW:^R  LOAD:^
    L  PRINT:^P ";: NORMAL
2400  IF C = 17 THEN  TEXT : END
    : REM CTRL-Q
```

New line to process a ctrl-P for hardcopy printout.

```
2500  IF C = 16 THEN  GOSUB 7000:
      REM CTRL-P
```

Routine to print the text file in memory to a printer.

```
6999  REM PRINTOUT ROUTINE
7000  HOME : VTAB 6:LIN = 0
```

Set up printing parameters.

```
7010  INPUT "LEFT MARGIN? (DEFAUL
    T = 10) ";X$:LM =  VAL (X$):
     IF LM < 1 THEN LM = 10
7020  PRINT : INPUT "RIGHT MARGIN
    ? (DEFAULT = 70) ";X$:RM =  VAL
    (X$): IF RM < 1 THEN RM = 70

7030  PRINT : INPUT "LINE SPACING
    ? (DEFAULT = 2) ";X$:LS =  VAL
    (X$): IF LS < 1 THEN LS = 2
7040 LL = RM - LM
7050  PRINT : PRINT "CONVERT TO L
    OWERCASE UNLESS PRECEDED
    BY @? (DEFAULT = NO) ";: GET
    X$:UC = 1: IF X$ = "Y" THEN
    UC = 0
```

Build a long string from stored lines of text until a carriage return is encountered or until the maximum string length is approached.

```
7060  PR# 1
7070  HOME : PRINT :P$ = "":CR =
    0:I = 0
7080 I = I + 1:P$ = P$ + L$(I)
7090  IF  RIGHT$ (P$,1) = CR$ THEN
    CR = 1: GOTO 7110
7100  IF  LEN (P$) < 255 - LWID AND
    I < LN THEN 7080
```

Go print some lines.

```
7110  GOSUB 7500:CR = 0
7120  IF I < LN THEN 7080
```

Print the last line.

```
7130  PRINT  TAB( LM);: IF UC THEN
     PRINT P$;: GOTO 7150
7140 PP$ = P$: GOSUB 60
7150  PRINT : PR# 0
```

Re-display the text on the screen.

```
7160  GOSUB 3000
7170  RETURN
```

Subroutine to print a string of text to the printer, breaking each line at the end of a word.

```
7500 L = LL
7510  IF  LEN (P$) > LL THEN 7550

7520  IF  NOT CR THEN 7640
7530 LP =  LEN (P$): IF LP < 2 THEN
    PP$ = "":P$ = "": GOTO 7590
7540 PP$ =  LEFT$ (P$,LP - 1):P$ =
    "": GOTO 7590
7550 C$ =  MID$ (P$,L,1): IF C$ =
    " " THEN 7580
7560 L = L - 1: IF L > 0 THEN 755
    0
7570 L = LL
7580 PP$ =  LEFT$ (P$,L):P$ =  RIGHT$
    (P$, LEN (P$) - L)
7590  PRINT  TAB( LM);: IF UC THEN
     PRINT PP$;: GOTO 7610
7600  GOSUB 60
7610  FOR J = 1 TO LS:LIN = LIN +
    1: PRINT " ": NEXT J
7615  IF LIN > 59 THEN  FOR J = 1
     TO 66 - LIN: PRINT : NEXT J
    :LIN = 0
7620  IF  LEN (P$) > LL THEN L =
    LL: GOTO 7550
7630  IF CR AND  LEN (P$) > 0 THEN
    7530
7640  RETURN
```

Subroutine to process errors without bombing the program.

```
19999  REM ERROR-HANDLING RTN
20000 E =  PEEK (222): PRINT  CHR$
    (7)
20010  IF E = 0 OR E > 15 THEN  PRINT
    "ERROR #";E;" IN LINE "; PEEK
    (218) +  PEEK (219) * 256: STOP

20020  PRINT D$;"CLOSE"
20030  PRINT "DISK ERR: ";
20040  IF E = 4 THEN  PRINT "WRIT
    E-PROTECTED";
20050  IF E = 5 THEN  PRINT "END
    OF DATA";
20060  IF E = 9 THEN  PRINT "DISK
     FULL";
20070  IF E = 10 THEN  PRINT "FIL
    E LOCKED";
20080  PRINT "; PRESS A KEY";: GET
    X$
20090  GOTO 210
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         ATARI BASIC           $
$       'MICROTEXT 1.1'         $
$     AUTHOR: JON VOSKUIL       $
$      (C) 1982  SOFTSIDE       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Lines to be changed or added to Microtext 1.0.

```
3 REM %     MICROTEXT 1.1     %
20 POSITION 8,8:PRINT "M I C R O T E X
 T   1 . 1"
100 PRINT CL$:TRAP 20000
125 DIM P$(255),PP$(80)
200 POSITION 2,0:PRINT "SAVE:^S  REVW:
^R   LOAD:^L   PRINT:^P";
2400 IF C=17 THEN END :REM CTL-Q
```

New line to process a ctrl-P for hardcopy printout.

```
2500 IF C=16 THEN GOSUB 7000:REM CTL-P
```

Routine to print the text file in memory to a printer.

```
6999 REM PRINTOUT RTN
7000 PRINT CL$:POSITION 2,6:LIN=0
```

Set up printing parameters.

```
7010 ? "Left margin? (Default = 10) ";
:INPUT X$:LM=10:IF LEN(X$)>0 THEN IF V
AL(X$)>0 THEN LM=VAL(X$):IF LM>37 THEN
 LM=37
7020 PRINT :PRINT "Right margin? (Defa
ult = 70) ";:INPUT X$:RM=70:IF LEN(X$)
>0 THEN IF VAL(X$)>0 THEN RM=VAL(X$)
7030 PRINT :PRINT "Line spacing? (Defa
ult = 2) ";:INPUT X$:LS=2:IF LEN(X$)>0
 THEN IF VAL(X$)>0 THEN LS=VAL(X$)
7040 LL=RM-LM
```

Build a string from stored lines of text until a carriage return is encountered or maximum dimensioned string length is approached.

```
7070 PRINT CL$:LPRINT "":P$="":CR=0:I=
0
7080 I=I+1:P$(LEN(P$)+1)=T$(LP(I-1)+1,
LP(I))
7090 IF P$(LEN(P$))=CR$ THEN CR=1:GOTO
 7110
7100 IF LEN(P$)<255-LWID AND I<LN-1 TH
EN 7080
```

Go print some lines.

```
7110 GOSUB 7500:CR=0
7120 IF I<LN-1 THEN 7080
```

Print the last line.

```
7130 LPRINT S$(1,LM);L$;
7150 LPRINT ""
```

Re-display the text on the screen.

```
7160 GOSUB 3000
7170 RETURN
```

Subroutine to print a string of text to the printer, breaking each line at the end of a word.

```
7500 L=LL
7510 IF LEN(P$)>LL THEN 7550
7520 IF ( NOT CR) THEN 7640
7530 LP=LEN(P$):IF LP<2 THEN PP$="":P$
="":GOTO 7590
7540 PP$=P$(1,LP-1):P$="":GOTO 7590
7550 C$=P$(L,L):IF C$=" " THEN 7580
7560 L=L-1:IF L>0 THEN 7550
7570 L=LL
7580 PP$=P$(1,L):P$=P$(L+1)
7590 LPRINT S$(1,LM);PP$;
7610 FOR J=1 TO LS:LIN=LIN+1:LPRINT ""
:NEXT J
7615 IF LIN>59 THEN FOR J=1 TO 66-LIN:
LPRINT "":NEXT J:LIN=0
7620 IF LEN(P$)>LL THEN L=LL:GOTO 7550
7630 IF CR AND LEN(P$)>0 THEN 7530
7640 RETURN
```

Subroutine to process errors without bombing the program.

```
19999 REM ERROR-HANDLING RTN
20000 CLOSE #2:E=PEEK(195)
20010 POSITION 2,0:PRINT S$;:POSITION
2,0:PRINT "Error: Code ";E;"; press an
y key";
20020 GET #1,X
20030 TRAP 20000
20040 GOTO 200
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ TRS-80 LEVEL II/DISK BASIC   $
$      'MICROTEXT 1.1'         $
$    AUTHOR: JON VOSKUIL       $
$     (C) 1982  SOFTSIDE       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

The following are lines to be changed or added to Microtext 1.0:

```
3 ' %     MICROTEXT 1.1     %
20 PRINT @198, "M I C R O T E X T   1 . 1"
50 FOR Z=1 TO 1000: NEXT Z: GOTO 100
```

Subroutine to print a line of upper-case text in lower case unless preceded by '@' to capitalize one letter or '@@' to capitalize whole word.

```
59 ' LOWERCASE PRINT RTN
60 X$="": IF PP$="" THEN 74
62 LOK=0: LC=-1: FOR K=1 TO LEN(PP$): CC=ASC(MID$(PP$,K,1)): IF
CC=32 THEN LOK=0
64 IF CC<>64 THEN 70
66 IF LC=0 THEN LOK=-1
68 LC=0: GOTO 72
70 LPRINT CHR$(CC-32* (LC AND CC>64 AND CC<91) * -(NOT LOK));;LC
=-1
72 NEXT
74 RETURN
```

A few miscellaneous changes.

```
125 ON ERROR GOTO 20000
200 PRINT @0, " SAVE: CLR-S     REVIEW: CLR-R     LOAD: CLR-L
  PRINT: CLR-P"
2400 IF C=81 THEN END: 'QUIT
```

New line to process a clear-P for hardcopy printout.

```
2500 IF C=80 THEN GOSUB 7000: GOTO 200: 'PRINT
```

Routine to print the text file in memory to a printer.

```
6999 ' PRINTOUT RTN
7000 CLS: LIN=0
```

Set up printing parameters.

```
7010 X$="10": INPUT"LEFT MARGIN (DEFAULT = 10) ";X$: LM=VAL(X$)
7020 PRINT: X$="70": INPUT"RIGHT MARGIN (DEFAULT = 70) ";X$: RM=
VAL(X$)
7030 PRINT: X$="2": INPUT"LINE SPACING (DEFAULT = 2) ";X$: LS=VA
L(X$)
7040 LL=RM-LM
7050 PRINT: X$="N": INPUT"CONVERT TO LOWERCASE, UNLESS PRECEDED
BY @ (DEFAULT = NO) ";X$: UC=-1: IF LEFT$(X$,1)="Y" THEN UC=0
```

Build a long string from stored lines of text until a carriage return is encountered or until the maximum string length is approached.

```
7070 CLS: LPRINT"": P$="": CR=0: I=0
7080 I=I+1: P$=P$+L$(I)
7090 IF RIGHT$(P$,1)=CR$ THEN CR=-1: GOTO 7110
7100 IF LEN(P$)<255-LWID AND I<LN THEN 7080
```

Go print some lines.

```
7110 GOSUB 7500: CR=0
7120 IF I<LN THEN 7080
```

Print the last line.

```
7130 LPRINT TAB(LM);: IF UC THEN LPRINT P$;: GOTO 7150
7140 PP$=P$: GOSUB 60
7150 LPRINT""
```

Re-display the text on the screen.

```
7160 GOSUB 3000
7170 RETURN
```

Subroutine to print a string of text to the printer, breaking each line at the end of a word.

```
7500 L=LL
7510 IF LEN(P$)>LL THEN 7550
7520 IF NOT CR THEN 7640
7530 LP=LEN(P$): IF LP<2 THEN PP$="": P$="": GOTO 7590
7540 PP$=LEFT$(P$,LP-1): P$="": GOTO 7590
7550 C$=MID$(P$,L,1): IF C$=" " THEN 7580
7560 L=L-1: IF L>0 THEN 7550
7570 L=LL
7580 PP$=LEFT$(P$,L): P$=RIGHT$(P$,LEN(P$)-L)
7590 LPRINT TAB(LM);: IF UC THEN LPRINT PP$;: GOTO 7610
7600 GOSUB 60
7610 FOR J=1 TO LS: LIN=LIN+1: LPRINT"": NEXT J
7615 IF LIN>59 THEN FOR J=1 TO 66-LIN: LPRINT"": NEXT J: LIN=0
7620 IF LEN(P$)>LL THEN L=LL: GOTO 7550
7630 IF CR AND LEN(P$)>0 THEN 7530
7640 RETURN
```

Subroutine to process errors without bombing the program.

```
19999 ' ERROR-HANDLING RTN
20000 PRINT@0,STRING$(63,32);
20010 E=ERR/2+1: PRINT@0, "ERROR:  CODE";E;
20050 PRINT"; PRESS ANY KEY";
20060 IF INKEY$="" THEN 20060
20070 P=P-1: RESUME 200
```

TRS-80® One Liners

```
1 CLS:A=RND(63)+128:FORX=0TO1023STEPRND(63):PRINT@X,CHR$(A);:OUT
255,4:OUT255,8:NEXTX:FORD=1TO200:NEXTD:RUN
```

Pat Hall
Holland, MI

```
1 CLS:PRINT"A*X[2+B*X+C=Y":INPUT"A=";A:INPUT"B=";B:INPUT"C=";C:C
LS:FORX=-32TO31STEP.5:Y=A*X[2+B*X+C:IFY<-24ORY>23THENNEXTX:FORI=
1TO2STEP0:IFINKEY$<>""THENRUNELSENEXTI:ELSESET(X*2+64,23-Y):NEXT
X:FORI=1TO2STEP0:IFINKEY$<>""THENRUNELSENEXTI
```

Paul Harper
Rimersburg, PA

REFLECTIVE SYMMETRY PATTERN SYSTEMS: VERTICAL REFLECTION

by Joan R. Truckenbrod

We are familiar with figures and objects that have been designed by using reflective symmetry, such as the front of a car or the front of most clothing. In these instances we can draw an invisible line down the center of the object and we find that both halves of the objects are exactly the same except that one side is the reverse or mirror image of the other side. Hold both of your hands out in front of you, palms up or down, and you see that one hand is the mirror image of the other. Your hands represent reflective symmetry. Hold one of your hands perpendicular to a mirror and you will see the same pair of hands. In these examples the axis of reflection is a vertical line and the figure is repeated sideways (in a horizontal direction). A graphics program that creates patterns based on horizontal reflections is described in the August, 1981, issue of ***SoftSide***.

In reflective symmetry figures can also be reflected, or mirrored, up or down (in a vertical direction). Using a horizontal axis of reflection, we can create an image in which the top half is the same as the bottom half with the bottom portion being a reflection of the top portion. In other words, the mirror is held in a horizontal position relative to the figure, and the figure is reflected up or down. A good example of vertical reflection is the reflection of buildings or people in a pond or lake in which the buildings appear upside down in the water. The letter F repeated in Figure 1 illustrates the reflection of a figure up and down.

Patterns can be created by repeating both of these reflective symmetry pairs individually or in combination. Two possible methods for creating patterns based on reflective symmetry are illustrated. The first pattern, shown in Figure 2, repeats the figure in its original orientation across the top row of the pattern. The next row of figures is the mirror image of the first row, using a horizontal axis of reflection. The following rows are each a vertical reflection of the previous row. Both types of vertical reflection are included in this pattern. Two examples of patterns created using this arrangement are shown in Figure 4. The pattern structure shown in Figure 3 uses vertical reflections also. The difference between the arrangement in Figures 2 and 3 is that the top row in Figure 3 consists of figures that are alternately right side up and upside down. Thus there is more visual variation within the pattern. Examples of patterns created with the ordering system shown in Figure 3, are illustrated in Figure 5. The program that creates a pattern based on the arrangement in Figure 2, and shown in Figure 4, is called Vertical Reflection 1. The program based on the arrangement shown in Figure 3 is listed as Vertical Reflection 2, and is further illustrated in the patterns in Figure 5.

The exciting aspect of working with patterns is that a wide range of patterns can be created with one patterning system by using different shapes or forms as the pattern module. Depending on the character and design of the pattern module, the patterns can appear to be very different. The figure or shape to be used as the pattern module is defined in X and Y coordinates within the range of 0 to 23. The program draws a shape by automatically connecting the points that define the shape. To allow you the freedom to use open or closed shapes, the program does not automatically connect the last point to the first point in the figure. In order to have a closed shape, the first point must also be considered to be the last point in the list of points that describe the figure. In line 60 of the programs you specify the number of points used to describe your figure. Then in line 95 the X and Y coordinates are listed sequentially in the DATA statement. Use graph paper with ten units per inch to draw your figure and to construct the list of points that define the figure.

Figure 1

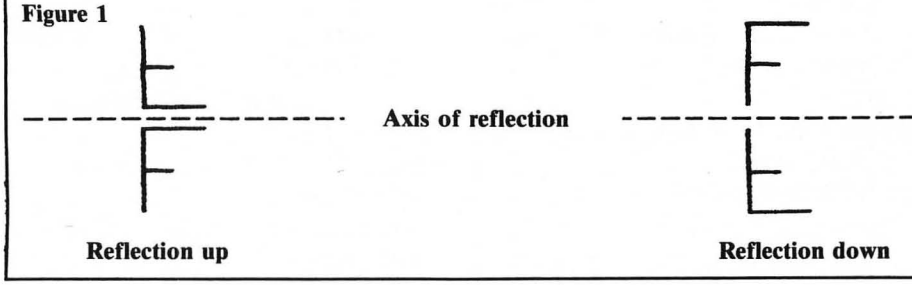
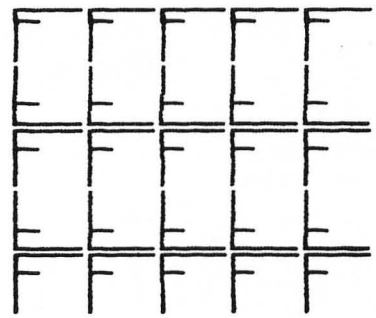
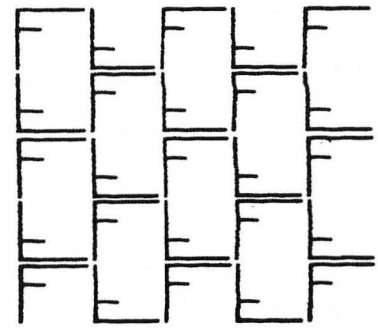


Figure 2



Vertical Reflection 1

Figure 3



Vertical Reflection 2

Figure 4

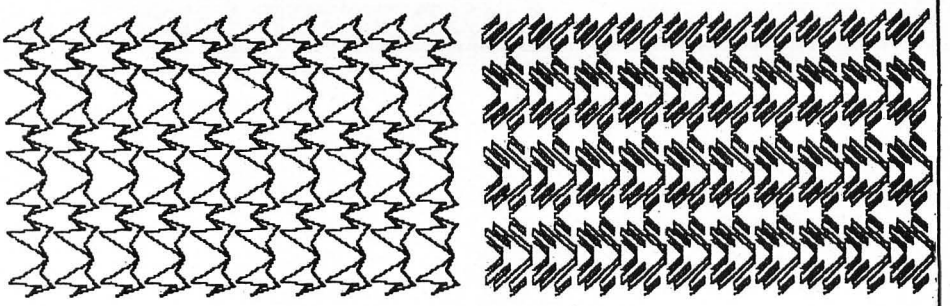
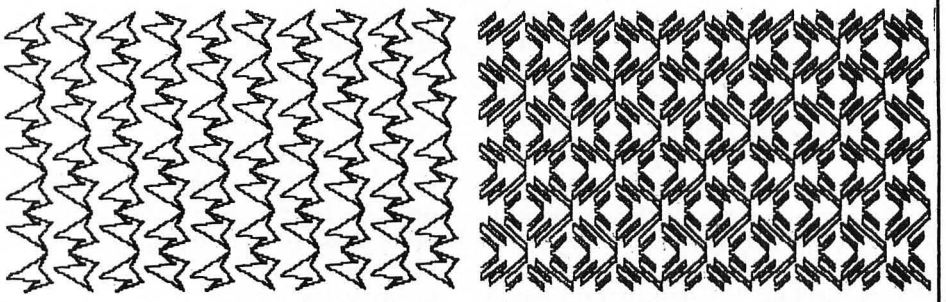


Figure 5

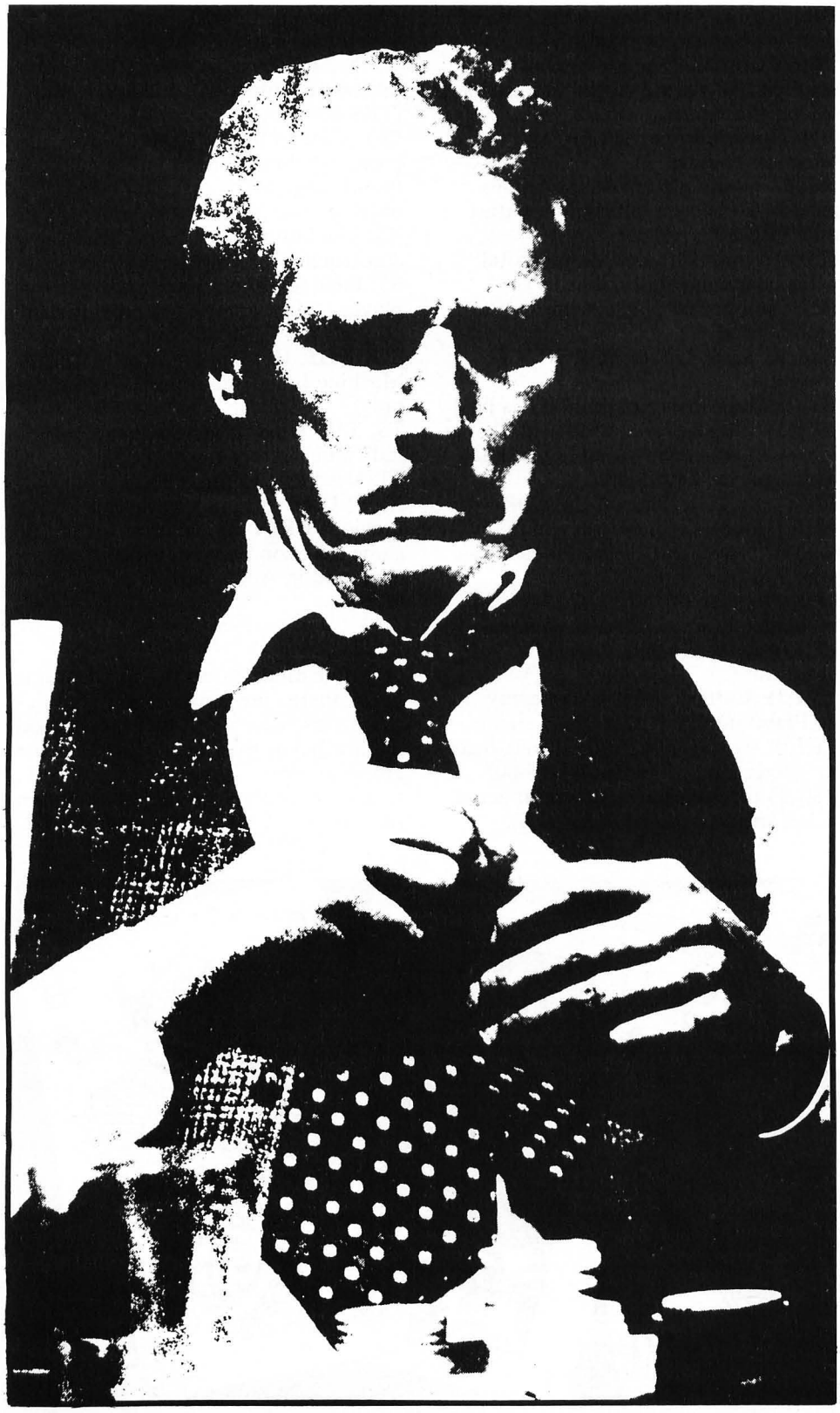


```
5  TEXT : HOME
10  REM VERTICAL REFLECTION 1
15  REM BY JOAN R. TRUCKENBROD
20  REM COPYRIGHT 1981
50  DIM X(50),Y(50),X3(50),Y3(50)
    ,NX(50),NY(50)
58  REM NPTS IS THE NUMBER OF
    POINTS IN THE FIGURE
60 NPTS = 12
70  FOR I = 1 TO NPTS: READ X(I),
    Y(I): NEXT I
90  REM DATA DESCRIBES THE FIGURE
95  DATA 15,0,23,10,23,17,20,23,1
    8,19,6,23,0,23,14,15,11,10,6
    ,10,0,5,15,0
103  GOSUB 4000
104  HGR2
105  HCOLOR= 7
107 L = 1
110  FOR N = 12 TO 160 STEP 24
120  FOR M = 5 TO 240 STEP 24
140  ON L GOSUB 5000,5300
150  GOSUB 3000
180  NEXT M
190 L = L + 1
200  IF L > 2 THEN L = 1
250  NEXT N
350  END
```

```
3000  REM PLOTTING SUBROUTINE
3005  IF N > 179 GOTO 350
3010  HPLOT NX(1) + M,NY(1) + N
3020  FOR I = 2 TO NPTS
3030  HPLOT  TO NX(I) + M,NY(I) +
      N
3040  NEXT I
3050  RETURN
4000  REM SUBROUTINE FOR VERTICAL
      REFLECTION
4020  FOR I = 1 TO NPTS
4030 X3(I) = X(I)
4040 Y3(I) = 23 - Y(I)
4045  NEXT I
4150  RETURN
4900  REM SUBROUTINE 5000 ASSIGNS
      VALUES FOR PLOTTING ARRAYS
      NX AND NY
5000  FOR I = 1 TO NPTS
5010 NX(I) = X(I)
5020 NY(I) = Y(I)
5030  NEXT I
5040  RETURN
5300  FOR I = 1 TO NPTS
5310 NX(I) = X3(I)
5320 NY(I) = Y3(I)
5330  NEXT I
5340  RETURN
```

```
5  TEXT : HOME
10  REM VERTICAL REFLECTION 2
15  REM BY JOAN R. TRUCKENBROD
20  REM COPYRIGHT 1981
50  DIM X(50),Y(50),X3(50),Y3(50)
    ,NX(50),NY(50)
58  REM NPTS IS THE NUMBER OF
    POINTS IN THE FIGURE
60 NPTS = 12
70  FOR I = 1 TO NPTS: READ X(I),
    Y(I): NEXT I
90  REM DATA DESCRIBES THE FIGURE
95  DATA 15,0,23,10,23,17,20,23,1
    8,19,6,23,0,23,14,15,11,10,6
    ,10,0,5,15,0
103  GOSUB 4000
104  HGR2
105  HCOLOR= 7
107 L = 1
109 C = 1
110  FOR N = 12 TO 160 STEP 24
120  FOR M = 5 TO 240 STEP 24
140  ON L GOSUB 5000,5300
150  GOSUB 3000
160 L = L + 1
170  IF L > 2 THEN L = 1
180  NEXT M
190 C = C + 1
200  IF C > 2 THEN C = 1
210 L = C
250  NEXT N
350  END
3000  REM PLOTTING SUBROUTINE
3005  IF N > 179 GOTO 350
3010  HPLOT NX(1) + M,NY(1) + N
3020  FOR I = 2 TO NPTS
3030  HPLOT  TO NX(I) + M,NY(I) +
      N
3040  NEXT I
3050  RETURN
4000  REM SUBROUTINE FOR VERTICAL
      REFLECTION
4020  FOR I = 1 TO NPTS
4030 X3(I) = X(I)
4040 Y3(I) = 23 - Y(I)
4045  NEXT I
4150  RETURN
4900  REM SUBROUTINE 5000 ASSIGNS
      VALUES FOR PLOTTING ARRAYS
      NX AND NY
5000  FOR I = 1 TO NPTS
5010 NX(I) = X(I)
5020 NY(I) = Y(I)
5030  NEXT I
5040  RETURN
5300  FOR I = 1 TO NPTS
5310 NX(I) = X3(I)
5320 NY(I) = Y3(I) + .5
5330  NEXT I
5340  RETURN
```

GAMBLER



by Randy Hawkins

***Gambler* is a game program for up to four players requiring a 24K Apple with Applesoft, a 32K Atari, or a 16K TRS-80® Model I or III. The TRS-80® version has an optional sound routine which requires an external amplifier.**

Gambler is more than a dozen different games of chance. You and as many as three of your friends compete in a contest to see who can parlay a $100 stake into $1000 first. The computer would not be content to just keep score and watch all this money change hands so it becomes a "player" with just the same chances of winning and losing as you have!

On each player's turn, he is first given the opportunity to buy as many as three lottery tickets. A lottery ticket will return $50 to the owner if it matches the winning numbers in the lottery held occasionally. After purchasing any tickets desired, the wheel of fortune is shown. A block of light dances randomly around the board stopping momentarily beside the name of the many games which can be played. The player touches any key and the dot slowly settles into one category — that is the game the player will participate in on his turn.

There are many types of games available in *Gambler*. Some are very simple; you may be instructed by a mysterious fortune teller to give or receive amounts of money to or from other players or the bank. The names of some of these categories are somewhat deceptive so pay close attention as the message crawls slowly across the screen.

There are also several betting games involving dice. You can win or lose money by betting on the outcome of the two dice rolls — will the total be even, will your roll be highest, or will it be the lucky number 7? There is even a form of "dice poker" with the best hand taking the pot.

The game of *Gambler* even has its own horse racing track so that all the players can bet on the outcome and try to win the purse. Lotteries are also held

once in a while to pay off the winning lottery tickets. Big money is available whenever a sweepstakes is held. The correct bet on the outcome of the roll of six dice can bring the winner $150, $200, $300, or even $450.

Gambler is loosely based on a board game with many extras added. Because the computer always plays, you can practice with it. And since as many as four humans can play along with the computer, it makes a great game for parties or when you have a large crowd eager to play with the computer. Overall, the program is fairly straightforward and easy to use.

The TRS-80® will make all its own decisions and even "touch keys" as directed by the program. The only exception to this is the note at the bottom of the page asking you to "TOUCH ANY KEY TO CONTINUE". Usually this delay is to allow the humans to read some instructions or make a decision. Even when it is the computer's turn, it will wait for you to give the signal to continue.

The winner is the first player to go over the $1000 mark. In case of a tie, the game continues until someone gets ahead. Should you go broke, the bank will advance you $100 but will charge you with one IOU. As soon as you are able, you must pay the IOU back at a price of $110. These transactions will take place only on your turn.

So the time has come. Type in this program, gather three of your friends around the video screen, and see who is the champion gambler in your household.

Variables

A$: Contains name of game read from the DATA statements in lines 390-410.
A1$, A2$: Graphics strings used as borders in game selection screen.
B$: Used for user response with INKEY$ function, or GET on the Apple. See variable B below for Atari equivalent.
D(6): Utility variables for six rolls of dice, six horse race positions, etc.
D$(6): Contains the pictures of the six dice. All six are stored in variable D$ on the Atari.
GN: Game number selected in selection routine.
H$: Contains picture of the horses used in horse race subroutine (Atari and TRS-80® only).
HM: Contains highest money total when searching for winner.
HN: Number of player with highest money total.
I, J, K, L, S: Utility FOR-NEXT counters.
IO(I): Contains number of IOUs held by player I.
IP: Used as a loop variable to signal which player's turn it is.
JJ: Upper boundary on the FOR-NEXT loop that increases gradually to slow movement of the dot in game selection function.
L(I): Number of lottery tickets held by player I (Atari only). Replaces L$(I,0) in Apple and TRS-80® program.
L$(I,J): Lottery ticket string array. I = Player number. Length of string L$(I,0) signals number of tickets held by player I on Apple and TRS-80®. L$(I,1) through L$(I,3) contain actual ticket strings. On the Atari, L$ contains all the ticket strings.
M(I): Contains amount of money help by player I.
M$: Used to hold the message printed out during various functions.
N: Contains the number of players.
N$(I): Contains name of player I. All names are contained in N$ in Atari version.
PB: Contains the number of IOUs which can presently be paid off.
Q: Dummy variable always used with the USR(0) function to create sound (TRS-80® only).
QQ: Used in lines 890 and 900 to create the four-tone "tune" played as the message M$ is printed (TRS-80® only).
R$: Used as a utility string in construction of graphics.
S$: First contains sweepstakes bet for player I, then contains results of that player's bet (TRS-80® only).
SOUND$: Holding area for sound Machine Language routine (TRS-80® only).
T$, T1$: Used to generate an individual lottery ticket.
TI: Used as a timing delay.
X, Y: Used to hold present and previous PRINT@ location for dot in game selection routine. X1 and Y1 also used in Apple and Atari versions.

One note on the Atari version: In some lines there are imbedded graphics characters, which will be explained on the line below them. If a character in a graphics string is not defined below, then it is simply a normal letter within the string.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      APPLESOFT BASIC        $
$         'GAMBLER'           $
$   AUTHOR: RANDY HAWKINS     $
$   TRANSL: RICH BOUCHARD     $
$     (C) 1982  SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

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

Program initialization. Line 20 creates A1$ and A2$, graphics strings used in the borders of the game selection screen. Subroutine 420 does further initialization of graphics. Lines 30 and 40 set the number of players, N. Line 50 determines players' names and initializes money, M(I), numbers of IOUs, IO(I), and lottery tickets, L$. If desired, instructions are displayed after prompting in line 50.

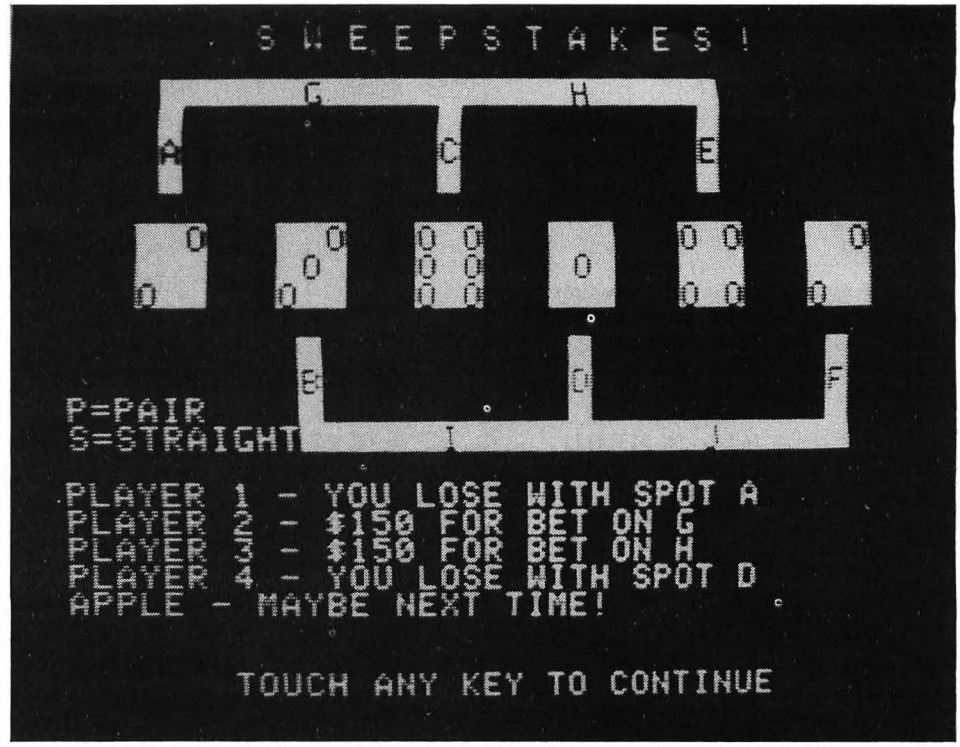
```
10  HOME : HTAB 14: PRINT "WELCOM
     E  TO": PRINT : HTAB 16: FLASH
     : PRINT "GAMBLER": NORMAL
15  DIM N$(6),M(6),D(6),IO(6),LO$
     (6,3),L$(5,3)
20 A2$ = "
                    ": GOSUB 420
     : GOSUB 1390
30  PRINT : PRINT "HOW MANY PLAYE
     RS (UP TO 4) ? ";
40  GET B$:N =  VAL (B$): IF N <
     1 OR N > 4 THEN 40
42  PRINT B$
50  FOR I = 1 TO N: PRINT "WHO IS
      PLAYER #";I;" ";: INPUT N$(
     I):M(I) = 100:IO(I) = 0:LO$(
     I,0) = "": NEXT I:N$(I) = "A
     PPLE":M(I) = 100:IO(I) = 0:N
      = N + 1: PRINT "DO YOU NEED
      INSTRUCTIONS ?";
60  GET B$: IF B$ = "Y" THEN  GOSUB
     1370
```

Game loop. Loop, using variable IP, cycles through each player's turn. Present money totals are shown in line 80. Problems dealing with players who have gone broke are handled in lines 90 through 120. Lottery tickets are bought in lines 130 through 150. Game number is selected in subroutine 290, and line 180 transfers control to the appropriate routine.

```
70  FOR IP = 1 TO N
80  HOME : INVERSE : PRINT "PRESE
     NT BANKROLLS
     IOUS   ";: NORMAL : PRINT "-
     ----------------------------
     ----------";: FOR J = 1 TO
     N: PRINT N$(J);: HTAB 20: PRINT
     M(J);: HTAB 35: PRINT IO(J):
      NEXT J: GOSUB 200
82  PRINT : PRINT : PRINT "IT IS
     ";N$(IP);"'S TURN.": IF M(IP
     ) >  = 0 THEN 100
90 IO(IP) = IO(IP) + 1:M(IP) = M(
     IP) + 100: IF M(IP) < 0 THEN
     90
92  PRINT "YOU MUST BORROW MONEY
     TO CONTINUE TO    PLAY AND P
     AY IT BACK TO THE BANK ...
      PLUS INTEREST!": FOR TI =
     1 TO 2000: NEXT TI: GOTO 80
100  IF IO(IP) = 0 OR M(IP) < 110
      THEN 130
110 PB =  INT (M(IP) / 110): IF P
     B > OI(IP) THEN PB = IO(IP)
120 IO(IP) = IO(IP) - PB:M(IP) =
     M(IP) - 110 * PB: PRINT "YOU
      CAN PAY BACK ";PB;" IOU NOT
     E(S) AT": PRINT "110 DOLLARS
      EACH.": FOR TI = 1 TO 2000:
      NEXT TI: GOTO 80
130  IF L$(IP,0) = "XXX" THEN 160
132  PRINT "WOULD YOU LIKE TO BUY
      A LOTTERY TICKET  FOR $10 ?
      ";
140  IF IP = N THEN  PRINT "Y": GOSUB
     260:L$(IP,0) = L$(IP,0) + "X
     ":L$(IP, LEN (L$(IP,0))) = T
     $:M(IP) = M(IP) - 10: FOR TI
      = 1 TO 1000: NEXT TI: GOTO
     80
150  GET B$: IF B$ = "N" THEN 160
152  IF B$ <  > "Y" THEN 150
154  PRINT "Y": GOSUB 260:L$(IP,0
     ) = L$(IP,0) + "X":L$(IP, LEN
     (L$(IP,0))) = T$:M(IP) = M(I
     P) - 10: FOR TI = 1 TO 1000:
      NEXT TI: GOTO 80
160  GOSUB 1470
170  GOSUB 290
180  ON GN GOSUB 1020,810,460,113
     0,1170,1190,460,1200,1230,12
     40,460,750,1300,940,1020,710
     ,810,1310,1320,750,460,1330,
     710,810,1340,1350,1330,810,4
     60,940,1310,710,750,1360,810
     ,710
190 IP = IP + 1: IF IP <  = N THEN
     80
192  GOTO 70
```

Called in game loop to check for winner. The highest money total is found in line 200. If greater than $1000, then winner is notified; otherwise game continues. Option of playing new game in lines 240 and 250.

```
200 HM = M(1):HN = 1: FOR J = 2 TO
     N: IF M(J) > HM THEN HM = M(
     J):HN = J
202  NEXT J
210  IF HM < 1000 THEN  RETURN
220 K = 0: FOR J = 1 TO N: IF M(J
     ) = HM THEN K = K + 1
222  NEXT J
230  IF K > 1 THEN  PRINT "SINCE
     THERE'S A TIE, KEEP GOING";:
      RETURN
240  PRINT : PRINT N$(HN);" WINS
     THIS GAME WITH": PRINT HM;"
```



```
    DOLLARS!"
242 FOR T = 1 TO 100: & TT,5: &
    T255 - T,5: NEXT T
246 PRINT "WOULD YOU LIKE TO PLA
    Y AGAIN ?";
250 GET B$: IF B$ = "Y" THEN  RUN

252 END
```

Lottery subroutine. Selects two-digit lottery ticket that is different from all other lottery tickets issued. Displays results in line 280 and returns.

```
260 T$ =  CHR$ (48 +  FN A(5)):T1
    $ =  CHR$ (48 +  FN A(6)): IF
    T1$ <  = T$ THEN 260
262 T$ = T$ + "-" + T1$: FOR I =
    1 TO N: IF L$(I,0) = "" THEN
    280
270 FOR J = 1 TO  LEN (L$(I,0)):
     IF T$ = L$(I,J) THEN 260
272 NEXT J
280 NEXT I: PRINT "YOU RECEIVE T
    ICKET NUMBER ";T$: FOR J = 1
     TO 20:A =  PEEK ( - 16336):
     NEXT J: RETURN
```

Game selector. Prints border using A1$ and A2$. Reads names of games from DATA statements in lines 390 through 410, and puts on screen. Lines 320 through 350 wait for players to touch any key. Lines 360 through 380 settle on one game, set the game number GN, and return.

```
290 RESTORE : HOME : INVERSE : PRINT
    A2$: FOR J = 1 TO 18: PRINT
    " ";: HTAB 3: PRINT " ";: HTAB
    20: PRINT " ";: HTAB 22: PRINT
    " ";: HTAB 39: PRINT " ": NEXT
    J: PRINT A2$: NORMAL
300 FOR X = 2 TO 19: READ A$: HTAB
    4: VTAB X: PRINT A$;: HTAB 2
    3: READ A$: PRINT A$;: NEXT
    X
310 POKE  - 16368,0:X1 = 2:X = 2
    : HTAB 1: VTAB 22: PRINT N$(
    IP);",": PRINT " TOUCH ANY K
    EY FOR YOUR SELECTION";
320 Y = X:Y1 = X1: INVERSE : HTAB
    X1: VTAB X: PRINT " ";: FOR
    T = X * 2 +  INT (X1 / 13) +
    10 TO X * 2 +  INT (X1 / 13)
     STEP  - 2: & TT,6: NEXT T:X
     =  FN A(18) + 1:X1 = ( FN A
    (2) - 1) * 19 + 2
330 IF IP = N AND  FN A(30) = 25
     THEN 360
332 IF IP = N THEN 350
340 IF  PEEK ( - 16384) > 128 THEN
     POKE  - 16368,0: GOTO 360
350 NORMAL : HTAB Y1: VTAB Y: PRINT
    " ";: GOTO 320
360 JJ = 100: FOR I = 1 TO 5 +  FN
    A(4): FOR J = 1 TO JJ: NEXT
    J:JJ = JJ + 100: HTAB Y1: VTAB
    Y: NORMAL : PRINT " ";:Y = X
    :Y1 = X1: INVERSE : HTAB X1:
     VTAB X: PRINT " ";: & TX *
    2 + X1 / 18 + 40,20:X1 = X1 +
    19: IF X1 > 22 THEN X1 = 2:X
     = X + 1
362 & TX * 2 + X1 / 18 + 30,20
370 IF X > 19 THEN X = 2
372 NEXT I
380 FOR TI = 1 TO 300: NEXT TI:G
    N = Y * 2 +  INT (Y1 / 16) -
    3: FOR I = 1 TO 5: HTAB Y1: VTAB
    Y: INVERSE : PRINT " ";: FOR
    T = 10 TO 120 STEP 10: & TT,
    5: NEXT T: NORMAL : HTAB Y1:
     PRINT " ";: FOR T = 110 TO
    10 STEP  - 10: & TT,5: NEXT
    T: NEXT I
390 DATA  POKER PARTY,F O R T U
     N E !,SWEEPSTAKES!,LOVE A N
    EIGHBOR,EASY COME EZ GO,WIN
    A FEW,SWEEPSTAKES!,UNLUCKY S
    EVEN,LOSE A FEW
400 DATA  EVEN STEVEN,SWEEPSTAKE
    S!,DAILY DOUBLE,POT LUCK,HIG
    H ROLLER,POKER PARTY,LOTTERY
    ,F O R T U N E !,JACKPOT,$10
    0 BONUS,OFF TO THE RACES,SWE
    EPSTAKES!,LOSE THIS TURN
410 DATA  LOTTERY,F O R T U N E
    !,TAX TIME,BONANZA,LOSE THIS
     TURN,F O R T U N E !,SWEEPS
    TAKES!,HIGH ROLLER,JACKPOT,L
    OTTERY,HORSE RACE,MAD MONEY,
    F O R T U N E !,LOTTERY
```

Graphics initialization. Set up graphics variables D$(1) through D$(6) which are pictures of the six dice.

```
420 R$ =  CHR$ (10) +  CHR$ (8) +
     CHR$ (8) +  CHR$ (8):D$(1) =
    "  ---- O ----  ":D$(2) =
    "  O----  ----O  "
430 D$(3) = "  O---- O ----O  ":D
    $(4) = "O O----  ----O O"
440 D$(5) = "O O---- O ----O O":D
    $(6) = "O O----O O----O O"
450 FOR I = 1 TO 6: FOR X = 4 TO
    11 STEP 7:D$(I) =  LEFT$ (D$
    (I),X - 1) + R$ +  MID$ (D$(
    I),X + 4): NEXT X: NEXT I: RETURN
455 HTAB 5 + X: PRINT " ";: HTAB
    17 + X: PRINT " ";: HTAB 29 +
    X: PRINT " ": RETURN
```

Sweepstakes routine. Lines 460 through 480 set up graphics display. Line 490 offers optional instructions, which are in lines 510 through 530. Lines 540 through 600 record each player's sweepstakes bet in S$(J). Line 620 rolls six dice, displays them, and records them in D(X). The highest number rolled is stored in HN. Line 630 checks for pairs, 640-650 for a straight, 660 for a split, and 670 for single spots.

```
460 X = 0: HOME : HTAB 9: PRINT "
    S W E E P S T A K E S !": PRINT
    : HTAB 5: INVERSE : PRINT "
        G           H      ": GOSUB
    455: HTAB 5: PRINT "A";: HTAB
    17: PRINT "C";:: HTAB 29: PRINT
    "E": GOSUB 455
462 VTAB 12:X = 6: GOSUB 455: HTAB
    11: PRINT "B";: HTAB 23: PRINT
    "D";: HTAB 35: PRINT "F": GOSUB
    455: HTAB 11: PRINT "      I
              J      "
470 HTAB 1: VTAB 14: NORMAL : PRINT
    "P=PAIR";: HTAB 1: VTAB 15: PRINT
    "S=STRAIGHT"
480 INVERSE : FOR X = 1 TO 6: HTAB
    X * 6 - 2: VTAB 8: PRINT D$(
    X);: NEXT X: NORMAL
490 HTAB 1: VTAB 17: PRINT "DO Y
    OU NEED SWEEPSTAKES INSTRUCT
    IONS ?";
500 GET B$
510 IF B$ = "N" THEN 540
512 HTAB 1: VTAB 17: CALL  - 958
    : PRINT "IN SWEEPSTAKES EVER
    YONE ANTES $10.  EACHPLAYER
    BETS ON THE OUTCOME OF THE R
    OLL  OF 6 DICE.  THERE ARE T
    HREE TYPES OF    BETS.": GOSUB
    1470
520 HTAB 1: VTAB 17: CALL  - 958
    : PRINT "ONE WAY IS TO BET O
    N WHERE THE HIGHEST  NUMBER
    ROLLED WILL APPEAR.  IF THE
    HIGH NUMBER APPEARS IN A,B,C
    ,D,E OR F YOU WIN$300.  LETT
    ERS G,H,I AND J COVER TWO
     SPOTS.  IF YOU BET G, YOU C
    OVER SPOTS A"
522 PRINT "AND C AND CAN WIN $15
    0."
530 GOSUB 1470: HTAB 1: VTAB 17:
     CALL  - 958: PRINT "ANOTHER
```

```
     BET IS <<P>> FOR PAIRS.  IF
      TWO DICE NEXT TO ONE ANOTHE
     R MATCH, YOU WIN $200.  THE
     THIRD BET IS <<S>> FOR A
      STRAIGHT.  IF 3 CONSECUTIVE
      DICE APPEAR IN NUMERICAL OR
     DER, YOU WIN $450."
532  GOSUB 1470
540  HTAB 1: VTAB 17: CALL  - 958
     : HTAB 30: PRINT "A-F $300";
      HTAB 30: PRINT "G-J $150"; HTAB
     32: PRINT "P $200"; HTAB 32:
      PRINT "S $450"; FOR I = 1 TO
     N:M(I) = M(I) - 10:S$(I) = "
     ": NEXT I:X = 17: VTAB 24: HTAB
     10: PRINT "PLACE YOUR BETS:"
     ;
550  FOR J = IP TO N: HTAB 1: VTAB
     X: PRINT N$(J);" - ";:X = X +
     1
560  IF J <  > N THEN 570
562 B$ =  CHR$ (64 +  FN A(11)): IF
     B$ = "K" THEN B$ = "P"
564  GOTO 580
570  GET B$: IF B$ > "@" AND B$ <
     "K" OR B$ = "P" OR B$ = "S" THEN
     580
572  GOTO 570
580  FOR K = 1 TO N: IF B$ = S$(K
     ) THEN 560
582  NEXT K: PRINT B$;:S$(J) = B$
     : NEXT J
590  IF IP = 1 THEN HN = 1: GOTO
     620
592  FOR J = 1 TO IP - 1: HTAB 1:
      VTAB X: PRINT N$(J);" - ";:
     X = X + 1
600  GET B$: IF B$ > "@" AND B$ <
     "K" OR B$ = "P" OR B$ = "S" THEN
     610
602  GOTO 600
610  FOR K = 1 TO N: IF B$ = S$(K
     ) THEN 600
612  NEXT K: PRINT B$;:S$(J) = B$
     : NEXT J:HN = 1
620  GOSUB 1470: INVERSE : FOR X =
     8 TO 10: HTAB 1: VTAB X: CALL
      - 868: NEXT X: FOR X = 1 TO
     6: FOR L = 1 TO 5:D(X) =  FN
     A(6): HTAB X * 6 - 2: VTAB 8
     : PRINT D$(D(X));: FOR TI =
     1 TO 50: NEXT TI: NEXT L: IF
     D(X) > HN THEN HN = D(X)
622  NEXT X: NORMAL
630  FOR J = 1 TO N: IF S$(J) <  >
     "P" THEN 640
632  FOR K = 1 TO 5: IF D(K) = D(
     K + 1) THEN S$(J) = "$200 FO
     R A PAIR":M(J) = M(J) + 200:
     K = 5: GOTO 690
634  NEXT K:S$(J) = "SORRY, NO PA
     IRS!": GOTO 690
640  IF S$(J) <  > "S" THEN 660
642  FOR K = 1 TO 4: IF D(K) = D(
     K + 1) - 1 AND D(K) = D(K +
     2) - 2 THEN S$(J) = "$450 FO
     R THE STRAIGHT":M(J) = M(J) +
     450: GOTO 690
650 S$(J) = "NO STRAIGHT!": GOTO
     690
660  IF (S$(J) = "G" AND (D(1) =
     HN OR D(3) = HN)) OR (S$(J) =
     "H" AND (D(3) = HN OR D(5) =
     HN)) OR (S$(J) = "I" AND (D(
     2) = HN OR D(4) = HN)) OR (S
     $(J) = "J" AND (D(4) = HN OR
     D(6) = HN)) THEN S$(J) = "$1
     50 FOR BET ON " + S$(J):M(J)
      = M(J) + 150: GOTO 690
670  IF S$(J) <  = "@" OR S$(J) >
      = "G" THEN 680
672 K =  ASC (S$(J)) - 64: IF D(K
     ) = HN THEN S$(J) = "$300 ON
      SPOT " + S$(J):M(J) = M(J) +
     300: GOTO 690
674 S$(J) = "YOU LOSE WITH SPOT "
      + S$(J)
680  IF  LEN (S$(J)) = 1 THEN S$(
     J) = "MAYBE NEXT TIME!"
690  NEXT J: HTAB 1: VTAB 17: CALL
      - 958:X = 17: FOR J = IP TO
     N: HTAB 1: VTAB X: PRINT N$(
     J);" - ";S$(J);:X = X + 1: NEXT
     J: IF IP <  > 1 THEN  FOR J =
     1 TO IP - 1: HTAB 1: VTAB X:
      PRINT N$(J);" - ";S$(J);:X =
     X + 1: NEXT J
700  GOSUB 1470: RETURN
```

Lottery subroutine. Instructions and lottery tickets shown in line 710. Array D(J) set equal to zero; six dice rolled and displayed. If a number J is rolled, D(J) set to 1. Winning tickets evaluated in line 730, lottery ticket string L$ set back to null and return.

```
710  HOME : HTAB 12: PRINT "L O T
      T E R Y": PRINT : PRINT "IF
      BOTH THE NUMBERS ON YOUR TI
     CKET ARE  AMONG THE SIX NUMB
     ERS ROLLED, YOU WILL  RECEIV
     E $50 FOR THAT TICKET.": PRINT
     "HERE WE GO ..."
712  FOR J = 1 TO N: HTAB 1: VTAB
     11 + J: PRINT N$(J);: HTAB 2
     0: PRINT L$(J,1);: HTAB 26: PRINT
     L$(J,2);: HTAB 32: PRINT L$(
     J,3): NEXT J: GOSUB 1470
720  INVERSE : FOR J = 1 TO 6:D(J
     ) = 0: NEXT J: FOR J = 1 TO
     6:K =  FN A(6): HTAB J * 6 -
     2: VTAB 8: PRINT D$(K);:D(K)
      = 1: NEXT J: NORMAL
```

```
730  FOR J = 1 TO N: IF L$(J,0) =
     "" THEN 738
732  FOR K = 1 TO  LEN (L$(J,0)):
      VTAB 11 + J: HTAB 14 + K *
     6: FOR L = 1 TO 550: NEXT L:
      IF D( VAL ( LEFT$ (L$(J,K),
     1))) = 1 AND D( VAL ( RIGHT$
     (L$(J,K),1))) = 1 THEN M(J) =
     M(J) + 50: PRINT "$50";: GOTO
     735
734  PRINT "---";: & T255,60: GOTO
     736
735  FOR T = 30 TO 10 STEP  - 2: &
     TT,10: & T255 - T,10: NEXT T

736  NEXT K
738  NEXT J
740  GOSUB 1470: FOR J = 1 TO N: FOR
     K = 0 TO 3:L$(J,K) = "": NEXT
     K: NEXT J: RETURN
745  & T250,3: PRINT "    ";: INVERSE
     : PRINT " ";: NORMAL : PRINT
     R$; CHR$ (8); CHR$ (8);" /";
     : INVERSE : PRINT "  ";: NORMAL
     : PRINT R$; CHR$ (8);"  (>";
     : & T245,6
747  & T255,3: PRINT  CHR$ (8); CHR$
     (8);")<";: RETURN
```

Horse race routine. Picture of a horse H$ in subroutine at line 745. Heading and instructions in lines 750 through 760. Horizontal screen position, in array D(K). Each of six horses is randomly advanced by incrementing print position. When D(K) approaches end of screen in line 780, winner announced in lines 790 and 800.

```
750  HOME : FOR J = 1 TO 6: HTAB
     2: VTAB J * 3: PRINT J;: HTAB
     11:D(J) = 11: GOSUB 745: VTAB
     J * 3: HTAB 39: PRINT ".";: NEXT
     J: FOR J = 1 TO N: HTAB 1: VTAB
     J * 3: PRINT N$(J);:M(J) = M
     (J) - 20: NEXT J: HTAB 5: VTAB
     1
752  PRINT "H O R S E   R A C E S
      !"
760  HTAB 1: VTAB 22: PRINT "EVER
     YONE HAS BET $20,": PRINT "T
     HE WINNER WILL WILL RECEIVE
     $100.": GOSUB 1470
770  REM
780  FOR I = 1 TO 3:K =  FN A(6):
     D(K) = D(K) + 1: HTAB D(K): VTAB
     K * 3: GOSUB 745: IF D(K) <
     35 THEN  NEXT I: GOTO 770
790  HTAB 1: VTAB 22: CALL  - 958
     : IF K <  = N THEN  PRINT N$
     (K);" ";:M(K) = M(K) + 100: GOTO
     800
792  PRINT "HORSE NUMBER ";K;" ";
800  PRINT "WINS THIS RACE!": GOSUB
     1470: RETURN
```

Fortune teller routine. Fortune teller's message chosen randomly in lines 810 through 880. Message displayed in lines 900 through 920.

```
810 M$ = "": HOME : VTAB 4: HTAB
     9: INVERSE : PRINT "THE FORT
     UNE TELLER SAYS:":K = 1:M =
     10 *  FN A(5) + 50
820  IF  FN A(5) = 5 THEN M$ = "H
     OLD A SWEEPSTAKES.":K = 2: GOTO
     890
830  IF  FN A(5) = 5 THEN M$ = "H
     OLD A LOTTERY.":K = 3: GOTO
     890
840  IF  FN A(2) = 2 THEN M$ = "C
     OLLECT FROM ":K =  - 1: GOTO
     850
842 M$ = "PAY TO "
850 J =  FN A(N + 1): IF J <  > I
     P THEN 860
852 M$ = M$ + "EVERYONE":M(IP) =
     M(IP) - K * N * M: FOR J = 1
      TO N:M(J) = M(J) + K * M: NEXT
     J: GOTO 880
860  IF J = N + 1 THEN M$ = M$ +
     "THE BANK":M(IP) = M(IP) - K
      * M: GOTO 880
870 M$ = M$ + N$(J):M(IP) = M(IP)
      - K * M:M(J) = M(J) + K * M

880 M$ = M$ + " " +  STR$ (M) + "
      DOLLARS."
890  REM
900 M$ = A2$ + N$(IP) + ", " + M$
      + A2$: FOR J = 1 TO  LEN (M
     $) - 38: & T(J -  INT (J / 4
     ) * 4) * 10 + 200,5: HTAB 1:
      VTAB 8: PRINT  MID$ (M$,J,4
     0);: FOR TI = 1 TO 55: NEXT
     TI: NEXT J:M$ = ""
910  NORMAL : GOSUB 1470
930  IF K = 2 THEN 460
932  IF K = 3 THEN 710
934  RETURN
```

High roller game. Instructions in line 940; all roll two dice in line 950. High roll determined in line 960 and 970. Single winner announced in line 980, tie roll causes reroll in lines 990 through 1010.

```
940  HOME : PRINT "H I G H   R O
     L L E R -- EVERYONE ANTES $2
     0 AND ROLLS TWO DICE.  THE H
     IGHEST    TOTAL TAKES THE $1
     00 PRIZE.  TO ROLL    DICE,
     TOUCH ANY KEY ON YOUR TURN."
     : GOSUB 1470
950  FOR J = 1 TO N:M(J) = M(J) -
     20:X = 3 + J * 3:X1 = (J -  INT
     (J / 2) * 2) * 3 + 12: HTAB
     1: VTAB X: PRINT N$(J);: GOSUB
     1450:D(J) = K:X1 = X1 + 6: GOSUB
     1450:D(J) = D(J) + K: HTAB 2
     5: VTAB X: PRINT "YOUR TOTAL
      IS ";D(J);: NEXT J
960 HN = D(1):HM = 1: FOR J = 2 TO
     N: IF D(J) > HN THEN HN = D(
     J):HM = J
962  NEXT J
970 K = 0: FOR J = 1 TO N: IF D(J
     ) = HN THEN K = K + 1
972  NEXT J
980  IF K = 1 THEN  HTAB 1: VTAB
     22: PRINT N$(HM);" WINS THE
     POT!":M(HM) = M(HM) + 100: GOSUB
     1470: RETURN
990  HTAB 1: VTAB 22: PRINT "WE'V
     E GOT A TIE!  THOSE HIGH ROL
     LERS    WILL ROLL AGAIN!": GOSUB
     1470: HTAB 1: VTAB 6: CALL  -
     958
1000  FOR J = 1 TO N: IF D(J) < H
     N THEN D(J) = 0: NEXT J: GOTO
     1010
1002 X = 3 + J * 3:X1 = (J -  INT
     (J / 2) * 2) * 3 + 12: HTAB
     1: VTAB X: PRINT N$(J);: GOSUB
     1450:D(J) = K:X1 = X1 + 6: GOSUB
     1450:D(J) = D(J) + K: HTAB 2
     5: VTAB X: PRINT "YOUR TOTAL
      IS ";D(J);: NEXT J
1010  GOTO 960
```

Poker party subroutine. Instructions in line 1020. Three dice rolled in lines 1030 for all players. Check for three of a kind in line 1040, pair in line 1050, and straight in line 1070. Winner announced in line 1120. If a tie, start routine again.

```
1020  HOME : PRINT "P O K E R   P
      A R T Y -- EACH PLAYER    P
     AYS $20 AND ROLLS THREE DICE
     .  THE BESTPOKER HAND (THREE
      OF A KIND > STRAIGHT >PAIR)
      WINS $100.": PRINT "TOUCH A
     NY KEY TO ROLL DICE.": GOSUB
     1470: FOR I = 1 TO N:M(I) =
     M(I) - 20: NEXT I:HM = 0
1030  FOR I = 1 TO N:X = I * 3 +
     4: HTAB 1: VTAB X: PRINT N$(
```

```
I);: FOR M = 1 TO 3:X1 = 10 +
M * 6:J = I: GOSUB 1450:D(M)
 = K: NEXT M:P(I) = 0: FOR T
I = 1 TO 350: NEXT TI: HTAB
16: VTAB X: CALL  - 958: PRINT
D(1);" ";D(2);" ";D(3);"  ";

1040  IF D(1) = D(2) AND D(1) = D
(3) THEN  PRINT "THREE OF A
KIND!";:P(I) = 30 + D(1): GOTO
1100
1050  IF D(1) = D(2) THEN P(I) =
10 + D(1)
1052  IF D(1) = D(3) THEN P(I) =
10 + D(1)
1054  IF D(2) = D(3) THEN P(I) =
10 + D(2)
1060  IF P(I) > 0 THEN  PRINT "A
PAIR!";: GOTO 1080
1070  FOR J = 1 TO 3: IF (D(1) =
D(2) + 1 AND D(1) = D(3) + 2
) OR (D(1) = D(2) - 1 AND D(
1) = D(3) - 2) THEN P(I) = 2
0: GOTO 1080
1072 HN = D(1):D(1) = D(2):D(2) =
D(3):D(3) = HN: NEXT J
1080 HN = D(1): FOR J = 2 TO 3: IF
D(J) > HN THEN HN = D(J)
1082  NEXT J
1090 P(I) = P(I) + HN: IF P(I) >
20 THEN  PRINT "STRAIGHT!";:
 GOTO 1100
1092  PRINT "HIGHEST ROLL = ";P(I
);
1100  IF P(I) > HM THEN HM = P(I)

1102  NEXT I
1110 K = 0: FOR I = 1 TO N: IF P(
I) = HM THEN K = K + 1:J = I
1112  NEXT I
1120  IF K = 1 THEN  HTAB 1: VTAB
22: PRINT N$(J);" WINS THE P
OT OF $100";M(J) = M(J) + 10
0: GOSUB 1470: RETURN
1122  HTAB 1: VTAB 22: PRINT "WE
HAVE A TIE...SO LET'S ALL PL
AY       ANOTHER HAND!": GOSUB
1470: HTAB 1: VTAB 6: CALL  -
958:HM = 0: GOTO 1030
```

Love thy neighbor routine. Set message in line 1150, adjust money in line 1160, and jump to display routine in line 890.

```
1130  HOME :K = 0:M =  FN A(5) *
10 + 50
1140 J =  FN A(N): IF J = IP THEN
1140
1150 M$ = "SHOW THAT YOU ARE A GO
OD NEIGHBOR AND GIVE " + N$(
J) + " " +  STR$ (M) + " DOL
LARS."
1160 M(IP) = M(IP) - M:M(J) = M(J
) + M: GOTO 890
```

Easy come easy go routine. Give instructions and roll two dice in line 1170. Adjust money and return.

```
1170  HOME : PRINT "E A S Y   C O
 M E   E A S Y   G O --   T
HE BANK WILL PAY YOU 10 TIME
S THE ROLL OF 2 DICE.  TOUCH
 ANY KEY TO ROLL DICE,": PRINT
N$(IP);".";: GOSUB 1470:J =
IP:X = 6:X1 = 14: GOSUB 1450
:M = K:X1 = 20: GOSUB 1450:M
= (M + K) * 10:M(IP) = M(IP
) + M
1172  HTAB 5: VTAB 12: PRINT "YOU
 WIN ";M;" DOLLARS."
1180  GOSUB 1470: RETURN
```

Short message routines. Simple procedures to handle following special routines: Win a Few (1190), Unlucky Seven (1200), Lose a Few (1230), Even Steven (1240), Pot Luck (1300), Jackpot (1310), $100 Bonus (1320), Lose a Turn (1330), Tax Time (1340), Bonanza (1350), and Mad Money (1360).

```
1190  HOME : PRINT "W I N   A   F
E W -- THE BANK WILL PAY  Y
OU TEN TIMES THE ROLL OF ONE
 DIE.": PRINT "TOUCH ANY KEY
 TO ROLL DIE, ";N$(IP);".": GOSU
1470:J = IP:X1 = 16:X = 6: GOSUB
1450:M = K * 10:M(IP) = M(IP
) + M: GOTO 1172
1200  HOME : PRINT "U N L U C K Y
  S E V E N -- ROLL TWO  D
ICE.  IF THE TOTAL IS SEVEN
YOU LOSE  $100.  FOR ANY OT
HER TOTAL, YOU WIN $100TOUCH
 ANY KEY TO ROLL DICE, ";N$(
IP);".": GOSUB 1470:J = IP:X
 = 6:X1 = 14
1202  GOSUB 1450:M = K:X1 = 20: GOSUB
1450:M = M + K
1210  IF M <  > 7 THEN M = 100:M(
IP) = M(IP) + M: GOTO 1172
1212 M = 100:M(IP) = M(IP) - M
1214  HTAB 5: VTAB 12: PRINT "YOU
 LOSE ";M;" DOLLARS.": GOSUB
1470: RETURN
```

```
1230  HOME : PRINT "L O S E   A
      F E W -- YOU MUST PAY THE B
      ANK TEN TIMES THE ROLL OF ON
      E DIE.": PRINT "TOUCH ANY KE
      Y TO ROLL DIE, ";N$(IP);".":
       GOSUB 1470:J = IP:X1 = 16:X
       = 6: GOSUB 1450:M = K * 10:
      M(IP) = M(IP) - M: GOTO 1214

1240  HOME : PRINT "E V E N   S T
      E V E N -- YOU MAY BET UP T
      O $90 AND ROLL TWO DICE.  IF
       THE TOTAL IS EVEN, YOU COLL
      ECT TWICE YOUR BET.": PRINT
      "TOUCH ANY KEY TO ROLL DICE.
      ": PRINT : PRINT N$(IP);", H
      OW MUCH WILL YOU BET ?";
1242  IF IP = N THEN B$ =  STR$ (
      FN A(4) + 5): GOTO 1260
1250  GET B$: IF  VAL (B$) < 1 THEN
      1250
1260 M =  VAL (B$) * 10: PRINT : HTAB
      22: PRINT M;" DOLLARS": GOSUB
      1470
1270 X = 9:X1 = 14:J = IP: GOSUB
      1450:P = K:X1 = 20: GOSUB 14
      50:P = P + K
1280  HTAB 5: VTAB 14: IF  INT (P
       / 2) * 2 = P THEN  PRINT "Y
      OU WIN ";2 * M;" DOLLARS.":M
      (IP) = M(IP) + 2 * M: GOTO 1
      290
1282  PRINT "YOU LOSE ";M;" DOLLA
      RS.":M(IP) = M(IP) - M
1290  GOSUB 1470: RETURN
1300  HOME :M = 10 *  FN A(5) + 1
      0:M(IP) = M(IP) - M:M$ = "WH
      Y DON'T YOU SWEETEN THE POT
      BY GIVING THE BANK " +  STR$
      (M) + " DOLLARS.":K = 0: GOTO
      890
1310  HOME :M =  FN A(5) * 10 + 5
      0:M$ = "YOU HIT THE JACKPOT!
        COLLECT " +  STR$ (M) + "
      DOLLARS.":K = 0:M(IP) = M(IP
      ) + M: GOTO 890
1320  HOME :M(IP) = M(IP) + 100:M
     $ = "PLEASE ACCEPT THIS $100
      BONUS!":K = 0: GOTO 890
1330  HOME :M$ = "YOU JUST LOST T
      HIS TURN!":K = 0: GOTO 890
1340  HOME :M$ = "PAY $100 TAX TO
       THE BANK!":M(IP) = M(IP) -
      100:K = 0: GOTO 890
1350  HOME :M =  FN A(5) * 10:M(I
      P) = M(IP) + M * N: FOR J =
      1 TO N:M(J) = M(J) - M: NEXT
      J:M$ = "WHAT A BONANZA!  EVE
      RYONE PAYS YOU " +  STR$ (M)
       + " DOLLARS.":K = 0: GOTO 8
      90
1360  HOME :M =  FN A(5) * 10:M(I
      P) = M(IP) - M * N: FOR J =
      1 TO N:M(J) = M(J) + M: NEXT
      J:M$ = "THIS OUGHT TO MAKE Y
      OU MAD.  GIVE EVERYBODY " +
       STR$ (M) + " DOLLARS.":K =
      0: GOTO 890
```

Game instructions. Called optionally from initialization section.

```
1370  HOME : PRINT "    THIS IS T
      HE GAME OF ";: FLASH : PRINT
      "GAMBLER!": NORMAL : PRINT :
       PRINT "YOU AND THE APPLE AR
      E IN CONTEST TO     SEE WHO
      WILL BUILD HIS $100 BANKROLL
          INTO $1000 FIRST.  MONEY
      IS MADE AND LOST";
1371  PRINT "THROUGH A SERIES OF
      GAMES OF CHANCE --"
1372  PRINT "FROM HORSE RACING AN
      D DICE GAMES, TO    LOTTERIE
      S AND SWEEPSTAKES!  IF YOU
          SHOULD LOSE ALL YOUR MON
      EY YOUR IOU WILLBE ACCEPTED
      (AS LONG AS YOU PAY IT BACK
      WITH INTEREST).  GOOD LUCK!"
      :K = 0: GOTO 910
1389  GOTO 1389
```

Utility subroutines. Lines 1390 through 1440 set up Ampersand Tones (from August, 1980, *SoftSide*, by Mark Cross). Lines 1450 through 1460 wait for input before stopping on one of the six dice pictures at screen position X1,X. The computer's dice choice is made in line 1450. Lines 1470 and 1480 print a message and wait for keyboard response.

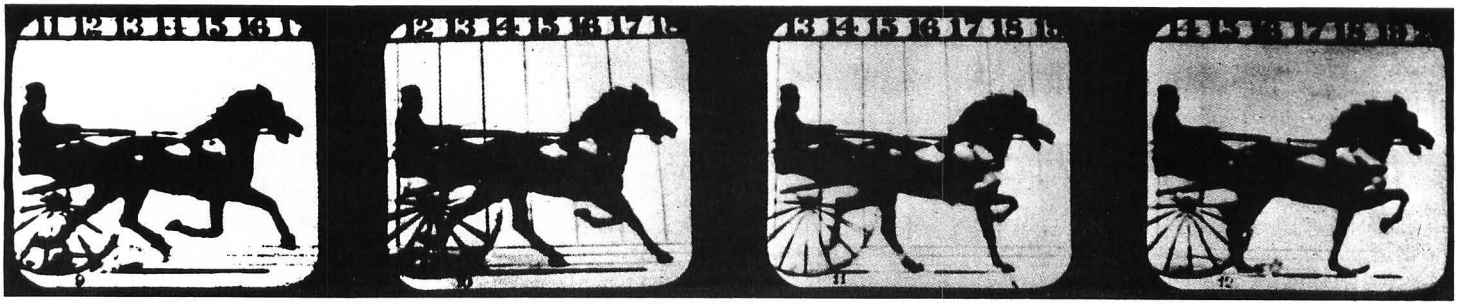
```
1390 J = 767:POK$ = "201,084,208,
      015,032,177,000,032,248,230,
      138,072,032,183,000,201,044,
      240,003,076,201,222,032,177,
      000,032,248,230,104,134,003,
      134,001,133,000": GOSUB 1430

1400 J = 802: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"
      : GOSUB 1430
1410  POKE 1013,76: POKE 1014,0: POKE
      1015,3: RETURN
1430  FOR I = 1 TO  LEN (POK$) /
      4 + 1: POKE I + J, VAL ( MID$
      (POK$,I * 4 - 3,3)): NEXT I:
       RETURN
1450  INVERSE : POKE  - 16336,0: IF
      J = N THEN  FOR L = 1 TO  FN
      A(50):K =  FN A(6): VTAB X: HTAB
      X1:A =  PEEK ( - 16336): PRINT
      D$(K);: NEXT L: NORMAL : RETURN

1460 K =  FN A(6): HTAB X1: VTAB
      X: PRINT D$(K);: IF  PEEK ( -
      16384) > = 128 THEN 1466
1462 A =  PEEK ( - 16336): GOTO 1
      460
1466  POKE  - 16368,0: NORMAL : RETUR

1470  HTAB 8: VTAB 24: PRINT "TOU
      CH ANY KEY TO CONTINUE";: POKE
       - 16368,0
1480  IF  PEEK ( - 16384) > = 12
      8 THEN  VTAB 24: HTAB 6: CALL
       - 868: POKE  - 16368,0: RETURN

1482  IF  FN A(9) = 1 THEN  & T FN
      A(150) + 50,5
1484  GOTO 1480
```



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         ATARI BASIC           $
$          'GAMBLER'            $
$    AUTHOR: RANDY HAWKINS      $
$    TRANSL: RICH BOUCHARD      $
$      (C) 1982  SOFTSIDE       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Program initialization. Line 20 creates A1$ and A2$, graphics strings used in the borders of the game selection screen. Subroutine 420 does further initialization of graphics. Lines 30 and 40 set the number of players, N. Line 50 determines players' names and initializes money, M(I), numbers of IOUs, IO(I), and lottery tickets, L$ and number of lottery tickets, L(I). If desired, instructions are displayed after prompting in line 50.

```
1 GRAPHICS 0
12 POKE 82,1:OPEN #2,4,0,"K:"
15 DIM N$(50),Z$(40),L$(50),M(6),IO(6)
,L(6),T$(3),T1$(3),A1$(40),A2$(40),A$(
30),D$(102),R$(4),O$(39),S$(6),B$(1)
16 DIM D(6),H$(16),M$(140),P(6)
18 FOR T=1 TO 50:N$(T,T)=" ":NEXT T:L$
=N$:O$=N$(1,39)
20 A1$="                    ":A1$(3,3)=
CHR$(25):A1$(LEN(A1$)+1)=A1$:A2$="___
_____________________________________
"
22 A1$(LEN(A1$)+1)="_":GOSUB 420
30 PRINT "How many players (up to 4) ?
 ";
40 GET #2,N:N=N-48:IF N<1 OR N>4 THEN
40
42 PRINT N
50 FOR I=1 TO N:? "Who is player #";I;
:INPUT Z$:N$(I*10-9,I*10)=Z$:M(I)=100:
IO(I)=0
52 L(I)=0:NEXT I:N$(I*10-9,I*10)="ATAR
I     ":M(I)=100:IO(I)=0:N=N+1:PRINT "
Do you need to see instructions?";
60 GET #2,B:IF CHR$(B)="Y" THEN GOSUB
1370
```

Game loop. Loop, using variable IP, cycles through each player's turn. Present money totals are shown in line 80. Problems dealing with players who have gone broke are handled in lines 90 through 120. Lottery tickets are bought in lines 130 through 150. Game number is *selected in subroutine 290*, and line 180 transfers control to the appropriate routine.

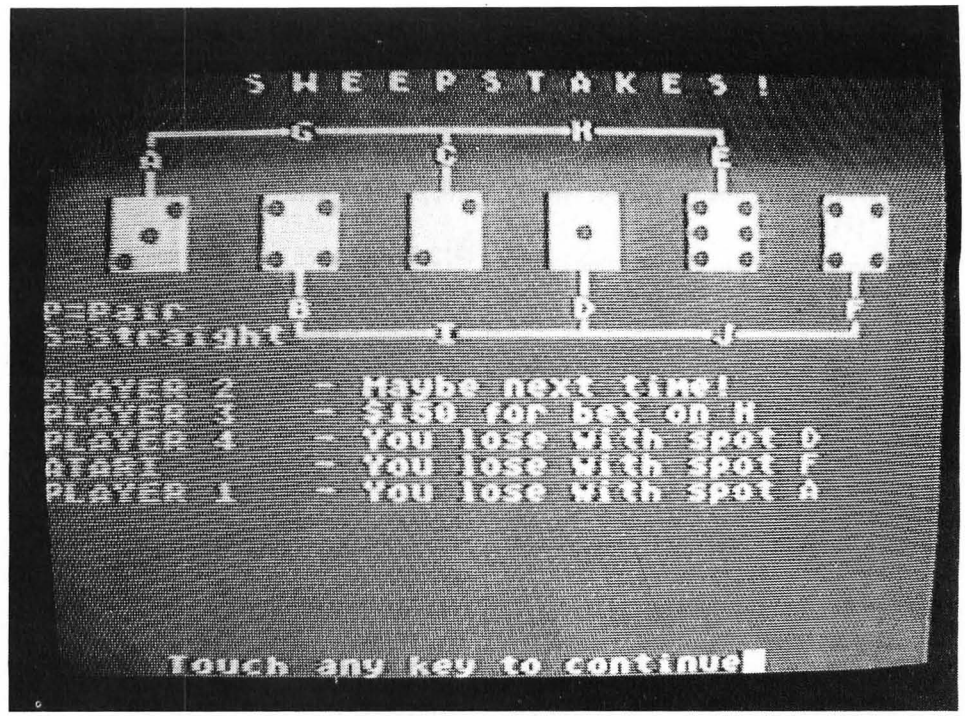
```
70 FOR IP=1 TO N
80 GRAPHICS 0:PRINT "Present bankrolls
              IOUs":? "
                      ";
81 REM Second string= 39 <CTRL> R's
82 FOR J=1 TO N:PRINT N$(J*10-9,J*10);
"          $";M(J):POSITION 35,J+1:? I
O(J):NEXT J:? :GOSUB 200
84 PRINT "It is ";N$(IP*10-9,IP*10);"'
s turn.":? :IF M(IP)>=0 THEN 100
90 IO(IP)=IO(IP)+1:M(IP)=M(IP)+100:IF
M(IP)<=0 THEN 90
92 PRINT "You must borrow money to con
tinue to  play and pay it back to the
 bank ....  plus interest!"
94 FOR TI=1 TO 800:NEXT TI:GOTO 80
100 IF IO(IP)<=0 OR M(IP)<110 THEN 130
110 PB=INT(M(IP)/110):IF PB>IO(IP) THE
N PB=IO(IP)
120 IO(IP)=IO(IP)-PB:M(IP)=M(IP)-110*P
B:PRINT "You can pay back ";PB;" IOU n
ote(s)":? "at 110 dollars each."
122 FOR TI=1 TO 800:NEXT TI
130 IF L(IP)=3 THEN 160
132 PRINT "Would you like to buy a lot
tery       ticket for $10 ? ";
140 IF IP=N THEN PRINT "Y":GOTO 154
150 GET #2,B:IF CHR$(B)="N" THEN 160
152 IF CHR$(B)<>"Y" THEN 150
154 ? CHR$(253):GOSUB 260:L(IP)=L(IP)+
1:L$(IP*9+L(IP)*3-11,IP*9+L(IP)*3-8)=T
$:M(IP)=M(IP)-10
156 FOR TI=1 TO 400:NEXT TI:GOTO 80
160 ? :POSITION 4,20:? "Hit any key to
 continue";
170 GET #2,B:GOSUB 290
180 ON GN GOSUB 1020,810,460,1130,1170
,1190,460,1200,1230,1240,460,750,1300,
940,1020,710,810,1310,1320,750,460
182 IF GN>21 THEN ON GN-21 GOSUB 1330,
710,810,1340,1350,1330,810,460,940,131
0,710,750,1360,810,710
190 IP=IP+1:IF IP>N THEN 70
192 GOTO 80
```

Called in game loop to check for winner. The highest money total is found in line 200. If greater than $1000, then winner is notified; otherwise game continues. Option of playing new game in lines 240 and 250.

```
200 HM=M(1):HN=1:FOR J=2 TO N:IF M(J)>
HM THEN HM=M(J):HN=J:NEXT J:GOTO 210
202 NEXT J
210 IF HM<1000 THEN RETURN
220 K=0:FOR J=1 TO N:IF M(J)=HM THEN K
=K+1
222 NEXT J
230 IF K>1 THEN PRINT "Since there's a
 tie, keep going":? :RETURN
240 ? :? N$(HN*10-9,HN*10);" wins this
 game with ";? HM;" dollars!"
242 FOR T=1 TO 100:SOUND 0,T,10,10:FOR
 TI=1 TO 5:NEXT TI:SOUND 0,255-T,10,10
:FOR TI=1 TO 5:NEXT TI:NEXT T
244 SOUND 0,0,0,0:SOUND 1,0,0,0
248 ? :? "Would you like to play again
 ?";
250 GET #2,B:IF CHR$(B)="Y" THEN RUN
252 END
```

Lottery subroutine. Selects two-digit lottery ticket that is different from all other lottery tickets issued. Displays results in line 280 and returns.

```
260 T$=" - ";T$(1,1)=CHR$(49+INT(RND(0
)*5)):T1$=CHR$(49+INT(RND(0)*6)):IF T1
$<=T$ THEN 260
```



```
262 T$(3,3)=T1$:FOR I=1 TO N:IF L(I)=0
 THEN 280
270 FOR J=1 TO L(I):IF T$=L$(I*9+J*3-1
1,I*9+J*3-9) THEN 260
272 NEXT J
280 NEXT I:PRINT "You receive ticket n
umber ";T$:RETURN
```

Game selector. Prints border using A1$ and A2$. Reads names of games from DATA statements in lines 390 through 410, and puts on screen. Lines 320 through 350 wait for players to touch any key. Lines 360 through 380 settle on one game, set the game number GN, and return.

```
290 RESTORE :GRAPHICS 0:PRINT A2$;:FOR
 J=1 TO 18:PRINT A1$;:NEXT J:PRINT A2$
;
300 FOR X=1 TO 18:READ A$:POSITION 4,X
:? A$:READ A$:POSITION 23,X:? A$:NEXT
X
310 X=1:X1=2:POKE 752,1:POKE 764,255:P
OSITION 2,22:? "Touch any key, ";N$(IP
*10-9,IP*10);
320 Y=X:Y1=X1:POSITION X1,X:? CHR$(160
);:X=INT(RND(0)*18)+1:X1=INT(RND(0)*2)
*19+2
322 FOR S=X*2+X1/21+50 TO X*2+X1/21+42
 STEP -4:SOUND 0,S,10,10:NEXT S:SOUND
0,0,0,0
330 IF IP=N AND RND(0)<0.03334 THEN 36
0
340 IF IP<>N THEN B=PEEK(764):IF B<>25
5 THEN 360
350 POSITION Y1,Y:? " ";:GOTO 320
360 POKE 764,255:JJ=25:FOR I=1 TO 6+IN
T(RND(0)*4):FOR J=1 TO JJ:NEXT J:JJ=JJ
+25:POSITION Y1,Y:? " ";:Y=X:Y1=X1
361 SOUND 0,X*2+X1/21+30,10,10:SOUND 0
,X*2+X1/21+40,10,10:SOUND 0,0,0,0
362 POSITION X1,X:? CHR$(160);:IF X1=2
 THEN X1=21:GOTO 370
364 X1=2:X=X+1:IF X=19 THEN X=1
370 NEXT I
380 FOR TI=1 TO 150:NEXT TI:GN=X*2+INT
(X1/13)-2:POSITION Y1,Y:? " ";:FOR I=1
 TO 5:POSITION Y1,Y:? CHR$(160);
381 FOR S=10 TO 120 STEP 10:SOUND 0,S,
10,10:NEXT S:FOR S=110 TO 0 STEP -10:S
OUND 0,S,10,10:NEXT S
382 FOR TI=1 TO 30:NEXT TI:POSITION Y1
,Y:? " ";:FOR TI=1 TO 30:NEXT TI:NEXT
I:POKE 752,0:RETURN
390 DATA POKER PARTY,F O R T U N E !,S
WEEPSTAKES!,LOVE A NEIGHBOR,EZ COME EZ
 GO,WIN A FEW
392 DATA SWEEPSTAKES!,UNLUCKY SEVEN,LO
SE A FEW
400 DATA EVEN STEVEN,SWEEPSTAKES!,DAIL
Y DOUBLE,POT LUCK,HIGH ROLLER,POKER PA
RTY,LOTTERY,F O R T U N E !
402 DATA JACKPOT,$100 BONUS,OFF TO THE
 RACES,SWEEPSTAKES!,LOSE THIS TURN
410 DATA LOTTERY,F O R T U N E !,TAX T
IME,BONANZA,LOSE THIS TURN,F O R T U N
 E !,SWEEPSTAKES!,HIGH ROLLER
412 DATA JACKPOT,LOTTERY,HORSE RACE,MA
D MONEY,F O R T U N E !,LOTTERY
```

Graphics initialization. Set up graphics variable D$ which is pictures of the six dice.

```
420 D$="   dlll T dlll     Tdlll   dll
lT    Tdlll T dlllT  T Tdlll   dlllT T
T Tdlll T dlllT TT Tdlll T Tdlll T"
430 REM Above: T=<CTRL> T
432 REM d=<ESC> <CTRL> =
434 REM l=<ESC> <CTRL> +
450 RETURN
```

Sweepstakes routine. Lines 460 through 480 set up graphics display. Line 490 offers optional instructions, which are in lines 510 through 530. Lines 540 through 600 record each player's sweepstakes bet in S$(J,J). Line 620 rolls six dice, displays them, and the records them in D(X). The highest number rolled is stored in HN. Line 630 checks for pairs, 640-650 for a straight, 660 for a split, and 670 for single spots.

```
460 GRAPHICS 0:? "         S W E E P S
T A K E S !":? :? "    QRRRRRGRRRRRWRR
RRRHRRRRE"
461 REM Above: Q=<CTRL> Q, R=<CTRL> R
462 REM W=<CTRL> W, E=<CTRL> E
464 ? "    A           C           E";
? "    !           !           !"
466 REM Above: !=<SHIFT> =
470 ? :? :? :? "          !
!          !";:? "P=Pair    B
  D          F"
471 REM Above: !=<SHIFT> =
472 ? "S=StraightZRRRRRIRRRRRXRRRRRJRR
RRRC"
473 REM Above: Z=<CTRL> Z, R=<CTRL> R
474 REM X=<CTRL> X, C=<CTRL> C
480 FOR X=1 TO 6:POSITION X*6-2,5:? D$
(X*17-16,X*17);:NEXT X
490 POSITION 1,12:? "Do you need Sweep
stakes instructions ?";:POKE 764,255
500 GET #2,B
510 IF CHR$(B)="N" THEN 540
512 POSITION 1,12:? "In Sweepstakes ev
eryone antes $10. Theneach player bets
 on the outcome of the"
514 ? "roll of 6 dice.  There are thre
e types of bets.":GOSUB 1470
520 POSITION 1,12:? "One way is to bet
 on where the highest number rolled wi
ll appear.  If the high";
522 ? "number appears in A,B,C,D,E or
F you   win $300.  Letters G,H,I and J
 cover   ";
524 ? "two spots.  If you bet G, you c
over    spots A and C and can win $150
."
530 GOSUB 1470:POSITION 1,12:? "Anothe
r bet is <<P>> for pairs.  If twodice
next to one another match, you win";
532 ? "$200.  The third bet is <<S>> f
or a    straight.  If 3 consecutive di
ce appear";
535 ? "in numerical order, you win $45
0.      ";D$:GOSUB 1470
540 FOR I=12 TO 18:POSITION 1,I:? D$;:
NEXT I:POSITION 30,12:? "A-F $300":POS
ITION 30,13:? "G-J $150"
542 POSITION 32,14:? "P $200":POSITION
 32,15:? "S $450":FOR I=1 TO N:M(I)=M(
I)-10:S$(I,I)=" ":NEXT I
544 POSITION 10,23:? "Place your bets:
";:X=12
550 FOR J=IP TO N:POSITION 1,X:? N$(J*
10-9,J*10);" - ";:X=X+1
560 IF J=N THEN B$=CHR$(65+RND(0)*11):
IF B$="K" THEN B$="P"
562 IF J=N THEN 580
570 GET #2,B:B$=CHR$(B):IF (B$)"@" AND
 B$<"K") OR B$="P" OR B$="S" THEN 580
572 GOTO 570
580 FOR K=1 TO N:IF B$=S$(K,K) THEN 56
0
582 NEXT K:PRINT B$;:S$(J,J)=B$:NEXT J
590 IF IP=1 THEN HN=1:GOTO 620
592 FOR J=1 TO IP-1:POSITION 1,X:? N$(
J*10-9,J*10);" - ";:X=X+1
600 GET #2,B:B$=CHR$(B):IF (B$)"@" AND
 B$<"K") OR B$="P" OR B$="S" THEN 610
602 GOTO 600
610 FOR K=1 TO N:IF B$=S$(K,K) THEN 60
0
612 NEXT K:PRINT B$;:S$(J,J)=B$:NEXT J
:HN=1
620 GOSUB 1470:POSITION 1,5:? D$(1,37)
:? D$(1,37):? D$(1,37):FOR X=1 TO 6:FO
R L=1 TO 5:D(X)=INT(RND(0)*6+1)
622 POSITION X*6-2,5:? D$(D(X)*17-16,D
(X)*17);:NEXT L:IF D(X)>HN THEN HN=D(X
)
624 NEXT X
628 FOR I=12 TO 15:POSITION 30,I:? "
      ";:NEXT I
630 FOR J=1 TO N:X=12+(J-IP):IF X<12 T
HEN X=X+N
631 POSITION 30,X:? "       ";:POSITI
ON 14,X:IF S$(J,J)<>"P" THEN 640
```


```
632 FOR K=1 TO 5:IF D(K)=D(K+1) THEN ?
 "$200 for a pair":M(J)=M(J)+200:K=5:G
OTO 690
634 NEXT K:? "Sorry, no pairs!":GOTO 6
90
640 IF S$(J,J)<>"S" THEN 660
642 FOR K=1 TO 4:IF D(K)=D(K+1)-1 AND
D(K)=D(K+2)-2 THEN ? "$450 for the str
aight":M(J)=M(J)+450:GOTO 690
644 NEXT K
650 ? "No straight!":GOTO 690
660 IF (S$(J,J)="G" AND (D(1)=HN OR D(
3)=HN)) OR (S$(J,J)="H" AND (D(3)=HN O
R D(5)=HN)) THEN 668
662 IF (S$(J,J)="I" AND (D(2)=HN OR D(
4)=HN)) OR (S$(J,J)="J" AND (D(4)=HN O
R D(6)=HN)) THEN 668
664 GOTO 670
668 ? "$150 for bet on ";S$(J,J):M(J)=
M(J)+150:GOTO 690
670 IF S$(J,J)<="@" OR S$(J,J)>="G" TH
EN 680
672 K=ASC(S$(J,J))-64:IF D(K)=HN THEN
PRINT "$300 on spot ";S$(J,J):M(J)=M(J
)+300:GOTO 690
674 PRINT "You lose with spot ";S$(J,J
);:GOTO 690
680 ? "Maybe next time!"
690 NEXT J
700 GOSUB 1470:RETURN
```

Lottery subroutine. Instructions and lottery tickets shown in line 710. Array D(J) set equal to zero; six dice rolled and displayed. If a number J is rolled, D(J) set to 1. Winning tickets evaluated in line 730, lottery ticket count L(I) set to zero and return.

```
710 GRAPHICS 0:? "          L O T T E
 R Y":? :? "If both the numbers on you
r ticket are"
712 ? "among the six numbers rolled, y
ou will receive $50 for that ticket.":
? "Here we go..."
714 FOR J=1 TO N:POSITION 1,10+J:? N$(
J*10-9,J*10);"     ";L$(J*9-8,J*9-6);"
   ";L$(J*9-5,J*9-3);"   ";L$(J*9-2,J*
9)
716 NEXT J:GOSUB 1470
720 FOR J=1 TO 6:D(J)=0:NEXT J:FOR J=1
 TO 6:K=INT(RND(0)*6)+1:POSITION J*6-2
,7:? D$(K*17-16,K*17):D(K)=1
722 NEXT J
730 FOR J=1 TO N:IF L(J)=0 THEN 738
731 FOR K=1 TO L(J):FOR L=1 TO 150:NEX
T L
732 IF D(VAL(L$(J*9+K*3-11,J*9+K*3-11)
))=1 AND D(VAL(L$(J*9+K*3-9,J*9+K*3-9)
))=1 THEN 735
734 POSITION 10+K*6,10+J:? "---";:SOUN
D 0,220,10,10:FOR TI=1 TO 100:NEXT TI:
GOTO 737
735 POSITION 10+K*6,10+J:? "$50";:M(J)
=M(J)+50:FOR S=30 TO 10 STEP -2:SOUND
0,S,10,10:FOR TI=1 TO 2:NEXT TI
736 SOUND 0,255-S,10,10:FOR TI=1 TO 2:
NEXT TI:NEXT S
737 SOUND 0,0,0,0:NEXT K:L(J)=0
738 NEXT J:FOR J=1 TO N*9:L$(J,J)=" ":
NEXT J
740 GOSUB 1470:FOR J=1 TO N*9:L$(J,J)=
" ":NEXT J:L(1)=0:L(2)=0:L(3)=0:RETURN
```

Horse race routine. Picture of a horse H$ in line 750. Heading and instructions in lines 750 through 760. Horizontal position in array D(K). Each of six horses is randomly advanced by incrementing print position. When D(K) approaches end of screen in line 780, winner announced in lines 790 and 800.

```
750 GRAPHICS 0:POKE 752,1:H$="
     _ _":FOR J=1 TO 6:D(J)=10:POSITIO
N 2,J*3+1:? J;"       ";H$;:NEXT J
751 REM 1st string: <SHIFT>' -  ' <CT
RL>U <ESC><SHIFT>= 5<ESC><SHIFT>+ <CTR
L>'  U '
752 FOR J=1 TO N:POSITION 1,J*3+1:PRIN
T N$(J*10-9,J*10);:M(J)=M(J)-20:NEXT J
:POSITION 6,0:? "H O R S E   R A C E S
 !"
754 FOR Y=1 TO 6:POSITION 38,Y*3+1:? "
.";:NEXT Y
760 POSITION 1,21:? "Everyone has bet
$20, the winner will  receive $100.":G
OSUB 1470
770 SOUND 0,250,8,10:SOUND 0,240,8,10:
SOUND 0,255,8,10:SOUND 0,0,0,0
780 K=INT(RND(0)*6)+1:D(K)=D(K)+1:POSI
TION D(K),K*3+1:? H$;:IF D(K)<33 THEN
770
790 POSITION 1,21:? O$(1,38);:? O$(1,3
8);:? O$(1,38);:POSITION 1,21:IF K>N T
HEN 794
792 ? N$(K*10-9,K*10);" ";:M(K)=M(K)+1
00:GOTO 800
794 SOUND 0,0,0,0:? "Horse number ";K;
" ";
800 ? "wins this race!":GOSUB 1470:POK
E 752,0:RETURN
```

Fortune teller routine. Fortune teller's message chosen randomly in lines 810 through 880. Message displayed in lines 900 through 920.

```
810 M$="":GRAPHICS 0:POSITION 7,4:? "I
HE FORTUNE TELLER SAYS:":K=1:M=10*INT(
RND(0)*5+1)+50
820 IF RND(0)<0.2 THEN M$="Hold a Swee
pstakes.":K=2:GOTO 890
830 IF RND(0)<0.2 THEN M$="Hold a Lott
ery.":K=3:GOTO 890
840 IF RND(0)<0.5 THEN M$="Collect fro
m ":K=-1:GOTO 850
842 M$="Pay to "
850 J=INT(RND(0)*(N+1)+1):IF J<>IP THE
N 860
852 M$(LEN(M$)+1)="everyone":M(IP)=M(I
P)-K*N*M:FOR J=1 TO N:M(J)=M(J)+K*M:NE
XT J:GOTO 880
860 IF J=N+1 THEN M$(LEN(M$)+1)="the b
ank":M(IP)=M(IP)-K*M:GOTO 880
870 M$(LEN(M$)+1)=N$(J*10-9,J*10):M(IP
)=M(IP)-K*M:M(J)=M(J)+K*M
880 M$(LEN(M$)+1)=" ":M$(LEN(M$)+1)=ST
R$(M):M$(LEN(M$)+1)=" dollars."
890 QQ=0
900 FOR TI=LEN(M$) TO 1 STEP -1:M$(TI+
44,TI+44)=M$(TI,TI):NEXT TI:M$(1,32)=0
$(1,32):M$(33,42)=N$(IP*10-9,IP*10)
901 M$(43,44)=", ":M$(LEN(M$)+1)=0$(1,
32)
902 FOR J=1 TO LEN(M$)-31:SOUND 0,200,
8,10:POSITION 5,6:? M$(J,J+30);:SOUND
0,0,0,0:FOR TI=1 TO 5:NEXT TI:NEXT J:M
$=""
910 GOSUB 1470
930 IF K=2 THEN 460
932 IF K=3 THEN 710
934 RETURN
```

High roller game. Instructions in line 940; all roll two dice in line 950. High roll determined in line 960 and 970. Single winner announced in line 980, tie roll causes reroll in lines 990 through 1010.

```
940 GRAPHICS 0:? "H I G H   R O L L E
R -- Everyone      antes $20 and rolls
 two dice.  The"
942 ? "highest roll takes the $100 pri
ze.  To roll dice, touch any key on yo
ur turn.":GOSUB 1470
950 FOR J=1 TO N:M(J)=M(J)-20:POSITION
 1,2+J*3:? N$(J*10-9,J*10):X1=12+(J-IN
T(J/2)*2)*4:X=2+J*3:GOSUB 1450:D(J)=K
952 X1=X1+8:GOSUB 1450:D(J)=D(J)+K:POS
ITION 28,X:? "Total is ";D(J);:NEXT J
960 HN=D(1):HM=1:FOR J=2 TO N:IF D(J)>
HN THEN HN=D(J):HM=J
962 NEXT J
970 K=0:FOR J=1 TO N:IF D(J)=HN THEN K
=K+1
972 NEXT J
980 IF K=1 THEN POSITION 1,20:? N$(HM*
10-9,HM*10);" wins the pot!":M(HM)=M(H
M)+100:POKE 764,255:GOSUB 1470:RETURN
990 POSITION 1,20:? "We've got a tie!
 Those high rollers  will roll again!
":POKE 764,255:GOSUB 1470
992 POSITION 1,5:FOR J=1 TO 18:? 0$;:N
EXT J
1000 FOR J=1 TO N:X1=14:X=2+3*J:IF D(J
)<HN THEN D(J)=0:NEXT J:GOTO 960
1002 POSITION 1,X:? N$(J*10-9,J*10):X1
=12+(J-INT(J/2)*2)*4:GOSUB 1450:D(J)=K
:X1=X1+8:GOSUB 1450
1004 D(J)=D(J)+K:POSITION 28,X:? "Tota
l is ";D(J);:X=X+3:NEXT J
1010 GOTO 960
```

Poker party subroutine. Instructions in line 1020. Three dice rolled in lines 1030 for all players. Check for three of a kind in line 1040, pair in line 1050, and straight in line 1070. Winner announced in line 1120. If a tie, start routine again.

```
1020 GRAPHICS 0:? "P O K E R   P A R T
 Y -- Each player  pays $20 and rolls
 three dice.  The"
1022 ? "best poker hand":? "(Three of
a Kind > Straight > Pair)   wins $100
.  Touch any key to roll dice.";
1024 GOSUB 1470:XX=6:FOR I=1 TO N:M(I)
=M(I)-20:NEXT I:HM=0
1030 FOR I=1 TO N:POSITION 1,XX:? N$(I
*10-9,I*10):FOR M=1 TO 3:J=I:X1=9+M*6:
X=XX:POKE 764,255:GOSUB 1450:D(M)=K
1032 NEXT M:P(I)=0
1040 FOR TI=1 TO 100:NEXT TI:FOR TI=X
TO XX+2:POSITION 11,TI:PRINT 0$(1,28);
:NEXT TI:POSITION 13,XX
1041 ? D(1);" ";D(2);" ";D(3);"  ";
1042 IF D(1)=D(2) AND D(1)=D(3) THEN ?
 "Three of a kind!";:P(I)=30+D(1):GOTO
 1100
1050 IF D(1)=D(2) THEN P(I)=10+D(1)
1052 IF D(1)=D(3) THEN P(I)=10+D(1)
1054 IF D(2)=D(3) THEN P(I)=10+D(2)
1060 IF P(I)>0 THEN PRINT "A pair!";:G
OTO 1100
1070 FOR J=1 TO 3:IF (D(1)=D(2)+1 AND
D(1)=D(3)+2) OR (D(1)=D(2)-1 AND D(1)=
D(3)-2) THEN P(I)=20:GOTO 1080
1072 HN=D(1):D(1)=D(2):D(2)=D(3):D(3)=
HN:NEXT J
1080 HN=D(1):FOR J=2 TO 3:IF D(J)>HN T
HEN HN=D(J)
1082 NEXT J
1090 P(I)=P(I)+HN:IF P(I)>20 THEN PRIN
T "A straight!";:GOTO 1100
1092 PRINT "Highest roll is a ";P(I);
1100 XX=XX+2:IF P(I)>HM THEN HM=P(I)
1102 NEXT I
1110 K=0:FOR I=1 TO N:IF P(I)=HM THEN
K=K+1:J=I
1112 NEXT I
1120 IF K=1 THEN POSITION 1,20:? N$(J*
10-9,J*10);" wins the pot of $100":M(J
)=M(J)+100:GOSUB 1470:RETURN
1122 POSITION 1,20:? "We have a tie ..
. so let's all play   another hand!";
GOSUB 1470:GOTO 1020
```

Love thy neighbor routine. Set message in line 1150, adjust money in line 1160, and jump to display routine in line 890.

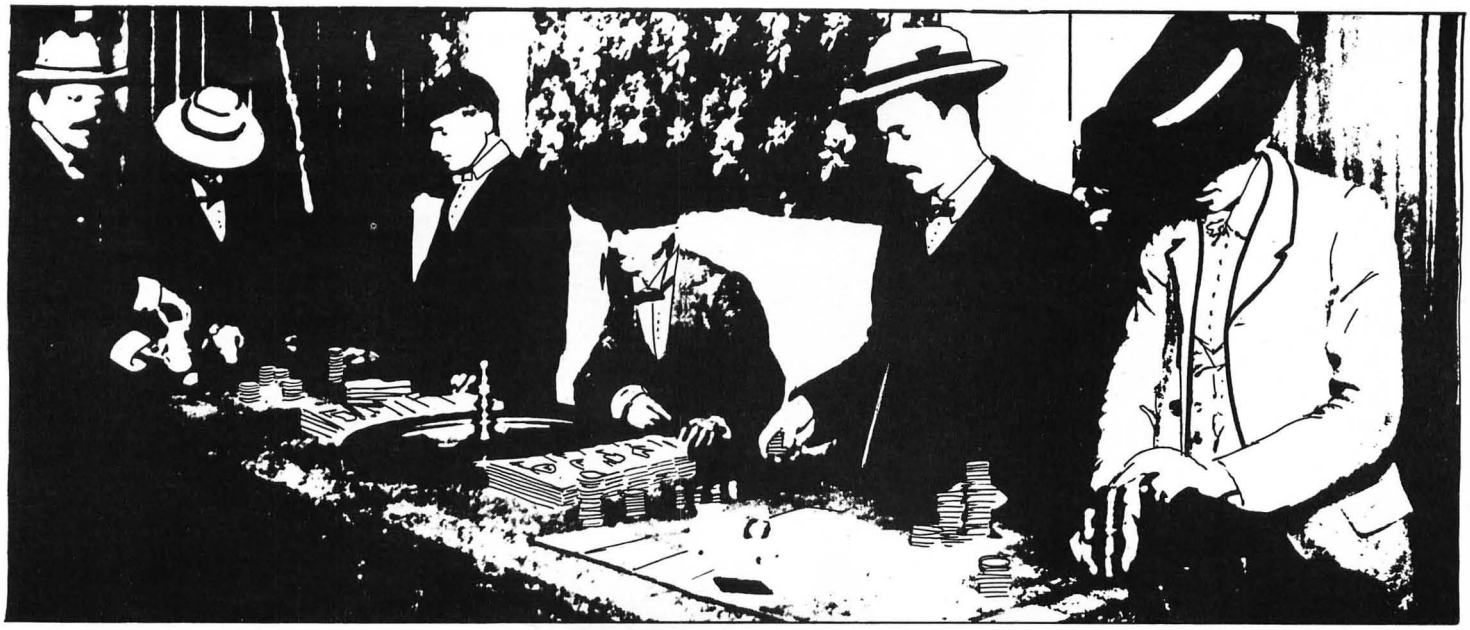
```
1130 GRAPHICS 0:K=0:M=INT(RND(0)*5+1)*
10+50
1140 J=INT(RND(0)*N+1):IF J=IP THEN 11
40
1150 M$="Show that you are a good neig
hbor and give ":M$(LEN(M$)+1)=N$(J*10-
9,J*10):M$(LEN(M$)+1)=" "
1152 M$(LEN(M$)+1)=STR$(M):M$(LEN(M$)+
1)=" dollars."
1160 M(IP)=M(IP)-M:M(J)=M(J)+M:GOTO 89
0
```

Easy come easy go routine. Give instructions and roll two dice in line 1170. Adjust money and return.

```
1170 GRAPHICS 0:? "E A S Y   C O M E
 E A S Y   G O  --  The bank will pay
you 10 times the roll";
1172 ? "of 2 dice.  Touch any key to r
oll dice,";N$(IP*10-9,IP*10):GOSUB 147
0:J=IP:X1=15:X=6:GOSUB 1450:M=K
1174 X1=X1+6:GOSUB 1450:M=(M+K)*10:M(I
P)=M(IP)+M
1176 POSITION 7,10:? "You win ";M;" do
llars."
1180 GOSUB 1470:RETURN
```

Short message routines. Simple procedures to handle following special routines: Win a Few (1190), Unlucky Seven (1200), Lose a Few (1230), Even Steven (1240), Pot Luck (1300), Jackpot (1310), $100 Bonus (1320), Lose a Turn (1330), Tax Time (1340), Bonanza (1350), and Mad Money (1360).

```
1190 GRAPHICS 0:? "W I N   A   F E W
-- The bank will payyou ten times the
roll of one die."
1192 ? "Touch any key to roll die, ";N
$(IP*10-9,IP*10):GOSUB 1470:J=IP:X1=18
:X=6:GOSUB 1450:M=K*10:M(IP)=M(IP)+M
1194 GOTO 1176
1200 GRAPHICS 0:? "U N L U C K Y   S E
 V E N  -- Roll two dice.  If the tota
l is seven you lose"
1202 ? "$100.  For any other total, yo
u win   $100.  Touch any key to roll
dice,":? N$(IP*10-9,IP*10):GOSUB 1470
1204 J=IP:X1=15:X=6:GOSUB 1450:M=K:X1=
X1+6:GOSUB 1450:M=M+K
```



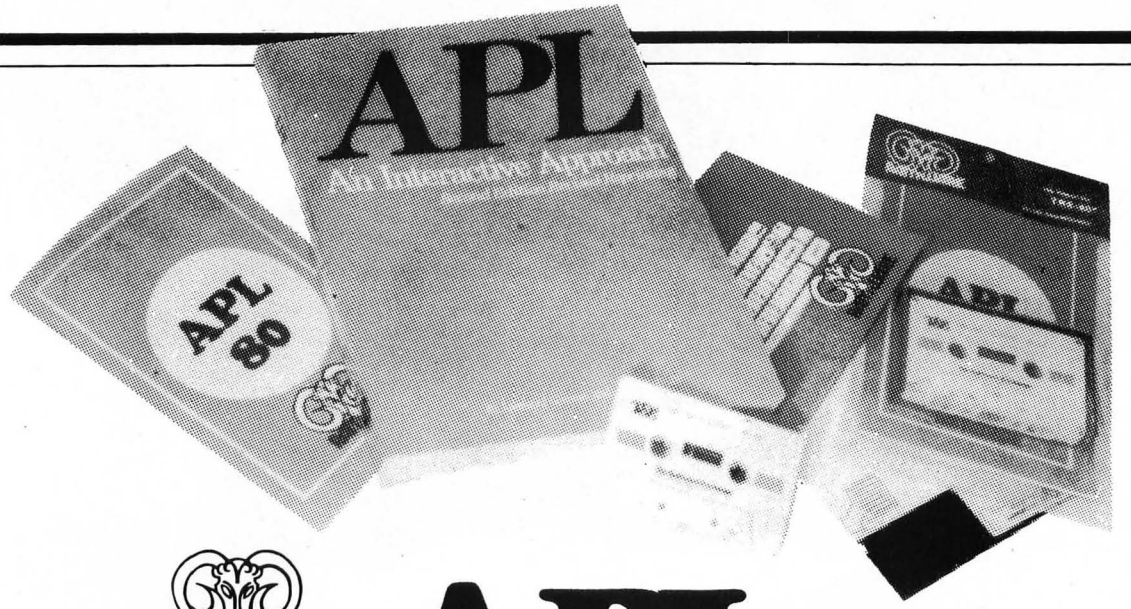
```
1210 POSITION 7,10:IF M=7 THEN ? "You
lose 100 dollars!":M(IP)=M(IP)-100:GOT
O 1220
1212 ? "You win 100 dollars!":M(IP)=M(
IP)+100
1220 GOSUB 1470:RETURN
1230 GRAPHICS 0:? "L O S E   A   F E W
  -- You must pay   the bank ten times
 the roll of one die.";
1232 ? "Touch any key to roll die,":?
N$(IP*10-9,IP*10):GOSUB 1470:J=IP:X1=1
8:X=6:GOSUB 1450:M=K*10
1234 M(IP)=M(IP)-M:POSITION 7,10:? "Yo
u lose ";M;" dollars.":GOSUB 1470:RETU
RN
1240 GRAPHICS 0:? "E V E N   S T E V E
 N  -- You may bet  up to $90 and roll
 two dice.  If the"
1242 ? "total is even, you collect twi
ce your  bet.  Touch any key to roll d
ice."
1244 ? N$(IP*10-9,IP*10):? "How much w
ill you bet ? ";:IF IP=N THEN B=54+INT
(RND(0)*4):GOTO 1260
1250 GET #2,B:IF B<49 OR B>57 THEN 125
0
1260 M=(B-48)*10:? M;" dollars":GOSUB
 1470
1270 X1=15:X=7:GOSUB 1450:P=K:X1=X1+6:
GOSUB 1450:P=P+K
1280 POSITION 7,11:IF INT(P/2)*2=P THE
N ? "You win ";2*M;" dollars.":M(IP)=M
(IP)+2*M:GOTO 1290
1282 ? "You lose ";M;" dollars.":M(IP)
=M(IP)-M
1290 GOSUB 1470:RETURN
1300 GRAPHICS 0:M=10*INT(RND(0)*5+1)+1
0:M(IP)=M(IP)-M:M$="Why don't you swee
ten the pot by giving the bank "
1302 M$(LEN(M$)+1)=STR$(M):M$(LEN(M$)+
1)=" dollars.":K=0:GOTO 890
1310 GRAPHICS 0:M=10*INT(RND(0)*5+1)+5
0:M$="You hit the Jackpot!  Collect ":
M$(LEN(M$)+1)=STR$(M)
1312 M$(LEN(M$)+1)=" dollars.":K=0:M(I
P)=M(IP)+M:GOTO 890
1320 GRAPHICS 0:M(IP)=M(IP)+100:M$="Pl
ease accept this $100 bonus!":K=0:GOTO
 890
1330 GRAPHICS 0:M$="You just lost this
 turn!":K=0:GOTO 890
1340 GRAPHICS 0:M$="Pay $100 tax to th
e bank!":M(IP)=M(IP)-100:K=0:GOTO 890
1350 GRAPHICS 0:M=INT(RND(0)*5+1)*10:M
(IP)=M(IP)+M*N:FOR J=1 TO N:M(J)=M(J)-
M:NEXT J
1352 M$="What a Bonanza!  Everyone pay
s you ":M$(LEN(M$)+1)=STR$(M):M$(LEN(M
$)+1)=" dollars.":K=0:GOTO 890
1360 GRAPHICS 0:M=INT(RND(0)*5+1)*10:M
(IP)=M(IP)-M*N:FOR J=1 TO N:M(J)=M(J)+
M:NEXT J
1362 M$="This ought to make you mad.
Give everybody ":M$(LEN(M$)+1)=STR$(M)
:M$(LEN(M$)+1)=" dollars.":K=0:GOTO 89
0
```

Game instructions. Called optionally from initialization section.

```
1370 GRAPHICS 0:? "This is the game of
 GAMBLER!":? :? "You and the Atari are
 in a contest to"
1372 ? "see who will build his $100 ba
nkroll  into $1000 first.  Money is m
ade and"
1374 ? "lost through a series of games
 of      chance -- from horse racing a
nd dice"
1380 ? "games to lotteries and sweepst
akes!   If you should lose all your m
oney, your"
1382 ? "IOU will be accepted (as long
as you  pay it back with interest).":
? :? "          GOOD LUCK!":K=0:GOTO
910
```

Utility subroutines. Lines 1450 through 1460 wait for input before stopping on one of the six dice pictures at screen position X1,X. The computer's dice choice is made in line 1450. Lines 1470 and 1480 print a message and wait for keyboard response.

```
1450 POKE 764,255:S=PEEK(752):POKE 752
,1
1452 IF J<>N THEN 1460
1454 FOR L=1 TO RND(0)*50+1:K=INT(RND(
0)*6+1):POSITION X1,X:? D$(K*17-16,K*1
7);:POKE 53279,RND(0)*4:NEXT L:GOTO 14
62
1460 B=PEEK(764):K=INT(RND(0)*6+1):POS
ITION X1,X:? D$(K*17-16,K*17);:POKE 53
279,RND(0)*4:IF B=255 THEN 1460
1462 POKE 764,255:POKE 752,S:RETURN
1469 STOP
1470 POKE 764,255:POSITION 1,23:? "
  Touch any key to continue";
1475 IF RND(0)<0.11 THEN S=RND(0)*70+3
0:FOR TI=S TO S-8 STEP -4:SOUND 0,TI,1
0,10:NEXT TI:SOUND 0,0,0,0
1480 IF PEEK(764)=255 THEN 1475
1482 POKE 764,255:POSITION 6,23:? "
                    ";:RETURN
```



RAINWARE

APL

by Phelps Gates

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

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

FEATURES

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

COMMANDS APL-80

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

SPECIFICATIONS

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

Reduced feature: 16K Level II tape version, no lessons.
Transpositions, format, and inner product not implemented. Reduced domain for some functions, 6 digit accuracy.
..... $14.95 on cassette

LIMITATIONS

Due to the absence of the special APL character set on the TRS-80 , APL-80 uses shifted letters to represent the various APL characters. In addition to the keyboard limitations, lamination, domino, and matris inverse are not implemented but can be derived with user-defined functions. Multiple specifications must be split into two statements unless the left-hand assignment is to a quad. This also applies to implied multiple specifications. Reduction and reshape (p) are not permitted for empty arguments; the argument of add/drop may not be scalar; empty indices are not permitted. A quad (q) can't be typed in response to a quad (nor can the name of a function which itself gets input from a quad). Quote-quad (m) is permitted. No more than 32 user functions can be defined in a single workspace and a function may not contain more than 255 lines.
A comment (c) must occupy a separate line: a comment can't follow a function statement on the same line.
In the tape version, arrays are limited to five (5) dimensions.

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

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ TRS-80 LEVEL II/DISK BASIC   $
$          'GAMBLER'           $
$    AUTHOR: RANDY HAWKINS     $
$      (C) 1982  SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Program initialization. Line 20 creates A1$ and A2$, graphics strings used in the borders of the game selection screen. Subroutine 420 does further initialization of graphics. Lines 30 and 40 set the number of players, N. Line 50 determines players' names and initializes money, M(I), numbers of IOUs, IO(I), and lottery tickets, L$. If desired, instructions are displayed after prompting in line 50.

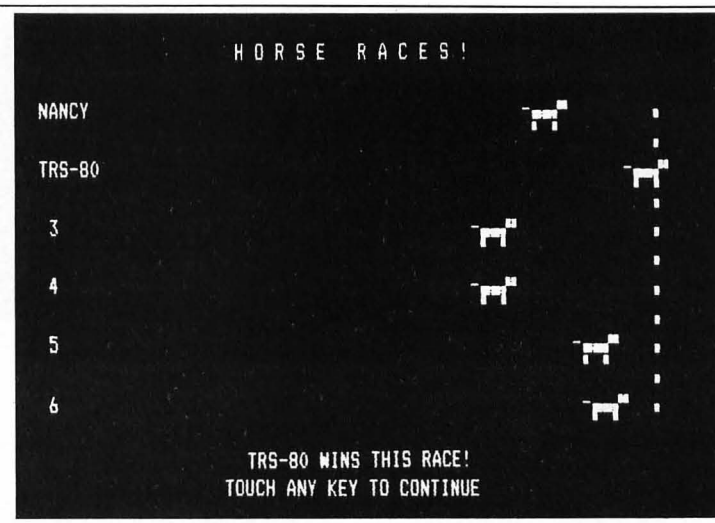
```
10 CLS:PRINTCHR$(23)"Welcome to *GAMBLER*
(with sound effects)":PRINT
20 CLEAR1000:DEFINTA-Z:A1$=CHR$(191)+"  "+CHR$(149)+STRING$(17,3
2):A1$=A1$+A1$+A1$+CHR$(191):A2$=CHR$(188)+STRING$(2,140)+CHR$(1
56)+STRING$(17,140):A2$=A2$+A2$+A2$+CHR$(188):GOSUB420:GOSUB1390
30 PRINT"How many players (up to 4) ? ";
40 B$=INKEY$:IFB$=""THEN40ELSEN=VAL(B$):IFN<1ORN>4THEN40ELSEPRIN
TN
50 FORI=1TON:PRINT"Who is player #";I;:INPUTN$(I):M(I)=100:IO(I)
=0:L$(I,0)="":NEXTI:N$(I)="TRS-80":M(I)=100:IO(I)=0:N=N+1:PRINT"

Do you need to see instructions?"
60 B$=INKEY$:IFB$=""THEN60ELSEIF(ASC(B$)AND95)=89THENGOSUB1370
```

Game loop. Loop, using variable IP, cycles through each player's turn. Present money totals are shown in line 80. Problems dealing with players who have gone broke are handled in lines 90 through 120. Lottery tickets are bought in lines 130 through 150. Game number is selected in subroutine 290, and line 180 transfers control to the appropriate routine.

```
70 FORIP=1TON
80 CLS:PRINTCHR$(23);"Present Bankrolls           IOUs";STRING$(
32,45);:FORJ=1TON:PRINTUSINGN$(J)+" "+STRING$(15-LEN(N$(J)),46)+
"$$###";M(J);:PRINT@122+J*64,IO(J);:NEXTJ:GOSUB200:PRINT"
It is ";N$(IP);"'s turn.":IFM(IP)>=0THEN100
90 IO(IP)=IO(IP)+1:M(IP)=M(IP)+100:IFM(IP)<0THEN90ELSEPRINT"You
must borrow money to cont-
inue to play and pay it back to
the bank ... plus interest!":FORTI=1TO2000:NEXTTI:GOTO80
100 IFIO(IP)>0ANDM(IP)>=110THENPB=INT(M(IP)/110)ELSE130
110 IFPB>IO(IP)THENPB=IO(IP)
120 IO(IP)=IO(IP)-PB:M(IP)=M(IP)-110*PB:PRINT"You can pay back";
PB;"IOU note
at 110 Dollars each.":FORTI=1TO2000:NEXTTI:GOTO80
130 IFL$(IP,0)="XXX"THEN160ELSEPRINT"Would you like to buy a lot
tery
ticket for $10 ? ";
140 IFIP=NTHENPRINT"Y":GOSUB260:L$(IP,0)=L$(IP,0)+"X":L$(IP,LEN(
L$(IP,0)))=T$:M(IP)=M(IP)-10:FORTI=1TO1000:NEXTTI:GOTO80
150 B$=INKEY$:IFB$=""THEN150ELSEB$=CHR$(ASC(B$)AND95):IFB$="N"TH
ENPRINT"N":GOTO160ELSEIFB$<>"Y"THEN150ELSEPRINT"Y":GOSUB260:L$(I
P,0)=L$(IP,0)+"X":L$(IP,LEN(L$(IP,0)))=T$:M(IP)=M(IP)-10:FORTI=1
TO1000:NEXTTI:GOTO80
160 PRINT@962,"Touch any key to continue";
170 B$=INKEY$:IFB$=""THEN170ELSEGOSUB290
180 ONGNGOSUB1020,810,460,1130,1170,1190,460,1200,1230,1240,460,
750,1300,940,1020,710,810,1310,1320,750,460,1330,710,810,1340,13
50,1330,810,460,940,1310,710,750,1360,810,710
190 NEXTIP:GOTO70
```

Called in game loop to check for winner. The highest money total is found in line 200. If greater than $1000, then winner is notified; otherwise game continues. Option of playing new game in lines 240 and 250.



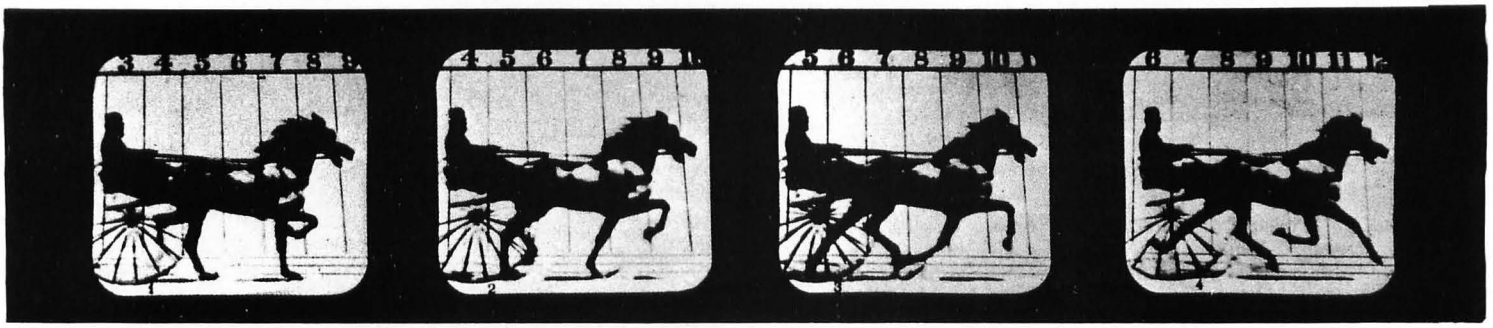

```
200 HM=M(1):HN=1:FORJ=2TON:IFM(J)>HMTHENHM=M(J):HN=J:NEXTJELSENE
XTJ
210 IFHM<1000THENRETURN
220 K=0:FORJ=1TON:IFM(J)=HMTHENK=K+1:NEXTJELSENEXTJ
230 IFK>1THENPRINT"Since there's a tie, keep going";:RETURN
240 PRINT:PRINT" ";N$(HN);" wins this game with
";HM;"dollars!":FORI=0TO90STEP2:Q=USR(11141-I):NEXTI:PRINTCHR$(2
3):PRINT"Would you like to play again ?"
250 B$=INKEY$:IFB$=""THEN250ELSEB$=CHR$(ASC(B$)AND95):IFB$="Y"TH
ENRUNELSEEND
```

Lottery subroutine. Selects two-digit lottery ticket that is different from all other lottery tickets issued. Displays results in line 280 and returns.

```
260 T$=CHR$(48+RND(5)):T1$=CHR$(48+RND(6)):IFT1$<=T$THEN260ELSET
$=T$+"-"+T1$:FORI=1TON:IFL$(I,0)=""THEN280
270 FORJ=1TOLEN(L$(I,0)):IFT$=L$(I,J)THEN260ELSENEXTJ
280 NEXTI:PRINT"You receive ticket number ";T$:FORJ=1TO20:OUT255
,9:OUT255,8:NEXTJ:RETURN
```

Game selector. Prints border using A1$ and A2$. Reads names of games from DATA statements in lines 390 through 410, and puts on screen. Lines 320 through 350 wait for players to touch any key. Lines 360 through 380 settle on one game, set the game number GN, and return.

```
290 RESTORE:CLS:PRINTA2$;:FORJ=1TO12:PRINTA1$;:NEXTJ:FORI=1TO
3:PRINTCHR$(143)STRING$(2,140)CHR$(141)STRING$(17,140);:NEXT:PRI
NTCHR$(143)
300 FORX=68TO772STEP64:READA$:PRINT@X,A$;:READA$:PRINT@X+21,A$;:
READA$:PRINT@X+42,A$;:NEXTX
310 X=65:B$=INKEY$:PRINT@970,N$(IP);", touch any key for your se
lection";
320 Y=X:PRINT@X,STRING$(2,143);:Q=USR(5000+RND(128)):X=RND(12)*6
4+RND(3)*21-20
330 IFIP=NTHENIFRND(30)=25THEN360ELSE350
340 B$=INKEY$:IFB$<>""THEN360
350 PRINT@Y,"  ";:GOTO320
360 JJ=100:KK=5160:FORI=1TO5+RND(4):Q=USR(KK):KK=KK+20:FORJ=1TOJ
J:NEXTJ:JJ=JJ+100:PRINT@Y,"  ";:Y=X:PRINT@X,STRING$(2,143);:X=X+
21:IFPOS(0)>40THENX=X+1
370 IFX>830THENX=65:NEXTIELSENEXTI
380 Q=USR(KK):FORTI=1TO300:NEXTTI:GN=INT(Y/21)-2:PRINT@Y,"  ";:F
ORI=1TO5:Q=USR(0):PRINT@Y,STRING$(2,143);:FORTI=1TO99:NEXTTI:PRI
NT@Y,"  ";:NEXTI:RETURN
390 DATAPoker Party,F O R T U N E !,Sweepstakes!,Love thy Neighb
or,Easy Come Easy Go,Win a Few,Sweepstakes!,Unlucky Seven,Lose a
 Few
```



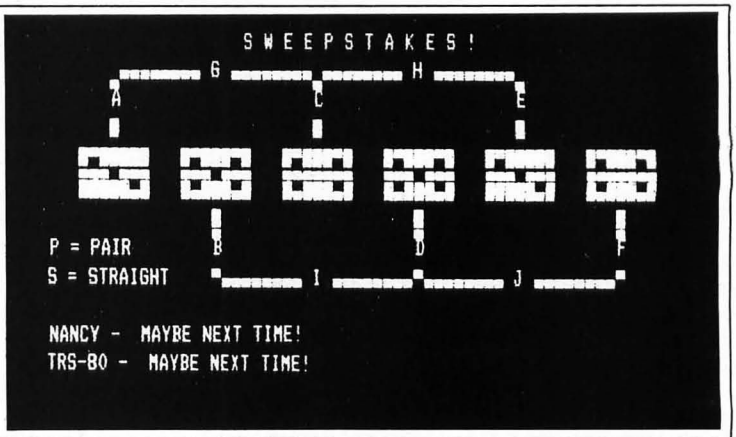
```
400 DATAEven Steven,Sweepstakes!,Daily Double,Pot Luck,High Roll
er,Poker Party,Lottery,F O R T U N E !,Jackpot,$100 Bonus,Off to
 the Races,Sweepstakes!,Lose This Turn
410 DATALottery,F O R T U N E !,Tax Time,Bonanza,Lose This Turn,
F O R T U N E !,Sweepstakes!,High Roller,Jackpot,Lottery,Horse R
ace,Mad Money,F O R T U N E !,Lottery
```

Graphics initialization. Set up graphics variables D$(1) through D$(6) which are pictures of the six dice.

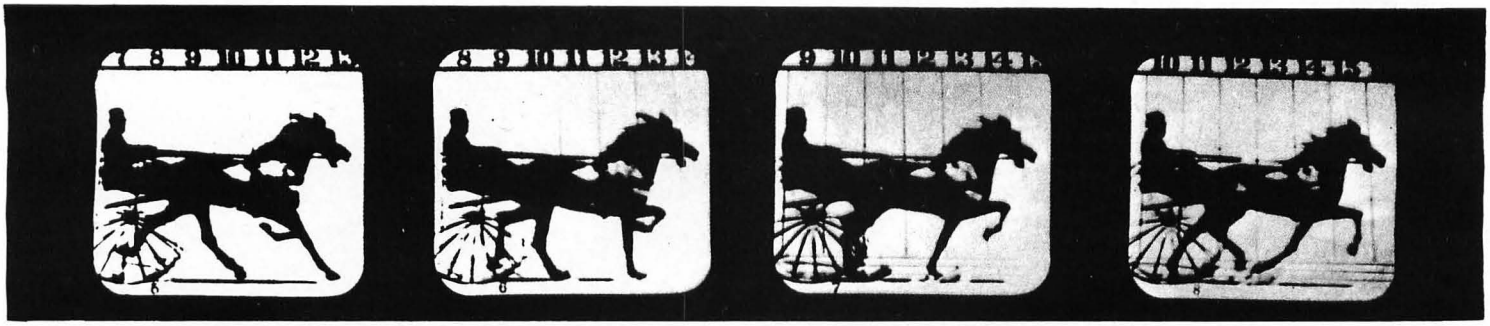
```
420 R$=CHR$(26)+STRING$(7,24):D$(1)=STRING$(3,191)+CHR$(143)+STR
ING$(3,191)+R$+STRING$(7,143):D$(2)=CHR$(191)+CHR$(179)+STRING$(
5,191)+R$+STRING$(5,143)+CHR$(140)+CHR$(143)
430 D$(3)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(143)+STRING$(3,191)
+R$+STRING$(5,143)+CHR$(140)+CHR$(143):D$(4)=CHR$(191)+CHR$(179)
+STRING$(3,191)+CHR$(179)+CHR$(191)+R$+CHR$(143)+CHR$(140)+STRIN
G$(3,143)+CHR$(140)+CHR$(143)
440 D$(5)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(143)+CHR$(191)+CHR$
(179)+CHR$(191)+R$+CHR$(143)+CHR$(140)+STRING$(3,143)+CHR$(140)+
CHR$(143):D$(6)=CHR$(191)+CHR$(179)+CHR$(191)+CHR$(179)+CHR$(191
)+CHR$(179)+CHR$(191)+R$+CHR$(143)+CHR$(140)+CHR$(143)
450 D$(6)=D$(6)+CHR$(140)+CHR$(143)+CHR$(140)+CHR$(143):RETURN
```

Sweepstakes routine. Lines 460 through 480 set up graphics display. Line 490 offers optional instructions, which are in lines 510 through 530. Lines 540 through 600 record each player's sweepstakes bet in S$(J). Line 620 rolls six dice, displays them, and records them in D(X). The highest number rolled is stored in HN. Line 630 checks for pairs, 640-650 for a straight, 660 for a split, and 670 for single spots. Results are displayed in line 690.

```
460 CLS:PRINT@19,"S W E E P S T A K E S !":PRINT@70,CHR$(176)STR
ING$(8,140)" G "STRING$(8,140)CHR$(176)STRING$(8,140)" H "STRING
$(8,140)CHR$(176):PRINT@134,"A":PRINT@154,"C":PRINT@174,"E":PRIN
T@198,CHR$(143)STRING$(19,32)CHR$(143)STRING$(19,32)CHR$(143)
470 PRINT@400,CHR$(191)STRING$(19,32)CHR$(191)STRING$(19,32)CHR$
(191):PRINT@464,"B":PRINT@484,"D":PRINT@504,"F":PRINT@528,CHR$(1
31)STRING$(8,140)" I "STRING$(8,140)CHR$(131)STRING$(8,140)" J "
STRING$(8,140)CHR$(131):PRINT@448,"P = PAIR";
480 PRINT@512,"S = STRAIGHT":FORX=1TO6:PRINT@249+X*10,D$(X);:NEX
TX
490 PRINT@650,"Do you need Sweepstakes instructions ?":B$=INKEY$
500 B$=INKEY$:IFB$<>""THEN510ELSEIFRND(9)=1THENQ=USR(1E4+RND(99)
):GOTO500ELSE500
510 B$=CHR$(ASC(B$)AND95):IFB$="N"THEN540ELSEPRINT@640,CHR$(31);
"In Sweepstakes everyone antes $10.  Each player bets on the out
-come of the roll of 6 dice.  There are three types of bets.":GO
SUB1470
520 PRINT@640,CHR$(31);"One way is to bet on where the highest n
umber rolled will appearIf the high number appears in A,B,C,D,E,
or F you win $300.  Let-ters G,H,I,and J cover two spots.  If yo
u bet G, you cover spotsA and C and can win $150."
```




```
530 GOSUB1470:PRINT@640,CHR$(31);"Another bet is <<P>> for pairs
.  If two dice next to one anothermatch, you win $200.  The thir
d bet is <<S>> for a straight.  If 3 consecutive dice appear i
n numerical order, you win $450.":GOSUB1470
540 PRINT@640,CHR$(31):PRINT@690,"A-F $300":PRINT@754,"G-J $150"
:PRINT@820,"P $200";PRINT@884,"S $450":FORI=1TON:M(I)=M(I)-10:S$
(I)="":NEXTI:X=640:PRINT@980,"Place your bets:";
550 FORJ=IPTON:PRINT@X,N$(J);" - ";:X=X+64
560 IFJ=NTHENB$=CHR$(64+RND(11)):IFB$="K"THENB$="P":GOTO580ELSE5
80
570 B$=INKEY$:IFB$=""THEN570ELSEB$=CHR$(ASC(B$)AND95):IF(B$>"@"A
NDB$<"K")ORB$="P"ORB$="S"THEN580ELSE570
580 FORK=1TON:IFB$=S$(K)THEN560ELSENEXTK:PRINTB$;:S$(J)=B$:NEXTJ
590 IFIP=1THENHN=1:GOTO620ELSEFORJ=1TOIP-1:PRINT@X,N$(J);" - ";:
X=X+64
600 B$=INKEY$:IFB$=""THEN600ELSEB$=CHR$(ASC(B$)AND95):IF(B$>"@"A
NDB$<"K")ORB$="P"ORB$="S"THEN610ELSE600
610 FORK=1TON:IFB$=S$(K)THEN600ELSENEXTK:PRINTB$;:S$(J)=B$:NEXTJ
:HN=1
620 GOSUB1470:PRINT@192,:PRINT@256,:FORX=1TO6:FORL=1TO5:Q=USR(5E
3+RND(128)):D(X)=RND(6):PRINT@249+X*10,D$(D(X));:NEXTL:IFD(X)>HN
THENHN=D(X):NEXTXELSENEXTX
630 FORJ=1TON:IFS$(J)="P"THENS$(J)="Sorry, no pairs!":FORK=1TO5:
IFD(K)=D(K+1)THENS$(J)=" $200 for a pair":M(J)=M(J)+200:K=5:NEXT
KELSENEXTK
640 IFS$(J)="S"THENFORK=1TO4:IFD(K)=D(K+1)-1ANDD(K)=D(K+2)-2THEN
S$(J)=" $450 for the straight":M(J)=M(J)+450:K=4:NEXTKELSENEXTK
650 IFS$(J)="S"THENS$(J)="No straight!":FORK=1TO4:IFD(K)=D(K+1)+
1ANDD(K)=D(K+2)+2THENS$(J)=" $450 for the straight":M(J)=M(J)+45
0:D=4:NEXTKELSENEXTK
660 IF(S$(J)="G"AND(D(1)=HNORD(3)=HN))OR(S$(J)="H"AND(D(3)=HNORD
(5)=HN))OR(S$(J)="I"AND(D(2)=HNORD(4)=HN))OR(S$(J)="J"AND(D(4)=H
NORD(6)=HN))THENS$(J)=" $150 for bet on "+S$(J):M(J)=M(J)+150
670 IFS$(J)>"@"ANDS$(J)<"G"THENK=ASC(S$(J))-64:IFD(K)=HNTHENS$(J
)=" $300 on spot "+S$(J):M(J)=M(J)+300ELSES$(J)=" You lose with
spot "+S$(J)
680 IFLEN(S$(J))=1THENS$(J)=" Maybe next time!"
```



```
690 NEXTJ:PRINT@576,CHR$(30):X=640:FORJ=1PTON:PRINT@X,N$(J);" -
";S$(J);CHR$(31):X=X+64:NEXTJ:IFIP=1THEN700ELSEFORJ=1TOIP-1:PRIN
T@X,N$(J);" - ";S$(J);:X=X+64:NEXTJ
700 GOSUB1470:RETURN
```

Lottery subroutine. Instructions and lottery tickets shown in line 710. Array D(J) set equal to zero; six dice rolled and displayed. If a number J is rolled, D(J) set to 1. Winning tickets evaluated in line 730, lottery ticket string L$ set back to null and return.

```
710 CLS:PRINT@24,"L O T T E R Y
If both the numbers on your ticket are among the six numbers
rolled, you will receive $50 for that ticket.  Here we go ...":F
ORJ=1TON:PRINT@576+J*64,N$(J)TAB(20)L$(J,1)TAB(30)L$(J,2)TAB(40)
L$(J,3):NEXTJ:GOSUB1470
720 FORJ=1TO6:D(J)=0:NEXTJ:FORJ=1TO6:K=RND(6):PRINT@313+J*10,D$(
K);:D(K)=1:Q=USR(1000*(J+1)):NEXTJ
730 FORJ=1TON:FORK=1TOLEN(L$(J,0)):FORL=1TO300:NEXTL:IFD(VAL(LEF
T$(L$(J,K),1)))=1ANDD(VAL(RIGHT$(L$(J,K),1)))=1THENM(J)=M(J)+50:
Q=USR(0):PRINT@586+J*64+K*10,"$50";:NEXTK,JELSEFORL=2032TO2080:Q
=USR(L):NEXTL:PRINT@586+J*64+K*10,"---";:NEXTK,J
740 GOSUB1470:FORJ=1TON:L$(J,0)="":FORK=1TO3:L$(J,K)="":NEXTK,J:
RETURN
```

Horse race routine. Picture of a horse H$ in line 750. Heading and instructions in lines 750 through 760. Screen print locations in array D(K). Each of six horses is randomly advanced by incrementing print position. When POS(0) passes 61 in line 780, winner announced in lines 790 and 800.

```
750 CLS:H$=CHR$(32)+"-"+CHR$(156)+CHR$(140)+CHR$(150)+CHR$(129):
FORJ=1TO6:D(J)=10+J*128:PRINT@J*128,J;TAB(10);H$;:NEXTJ:FORJ=1TO
N:PRINT@J*128,N$(J);:M(J)=M(J)-20:NEXTJ:PRINT@19,"H O R S E   R
A C E S !";:FORY=7TO37STEP3:SET(120,Y):NEXTY
760 PRINT@900,"Everyone has bet $20, the winner will receive $10
0.":GOSUB1470
770 FORJ=1TO3:Q=USR(266):FORK=1TO20:NEXTK,J
780 FORI=1TO3:K=RND(6):D(K)=D(K)+1:PRINT@D(K),H$;:IFPOS(0)>61THE
NI=3:NEXTELSENEXT:GOTO770
790 PRINT@896,CHR$(31);:IFK<=NTHENPRINTTAB(23-LEN(N$(K))/2);N$(K
);" ";:M(K)=M(K)+100ELSEPRINTTAB(16);"Horse Number";K;
800 PRINT"wins this race!":GOSUB1470:RETURN
```

Fortune teller routine. Fortune teller's message chosen randomly in lines 810 through 880. Message displayed in lines 900 through 920.

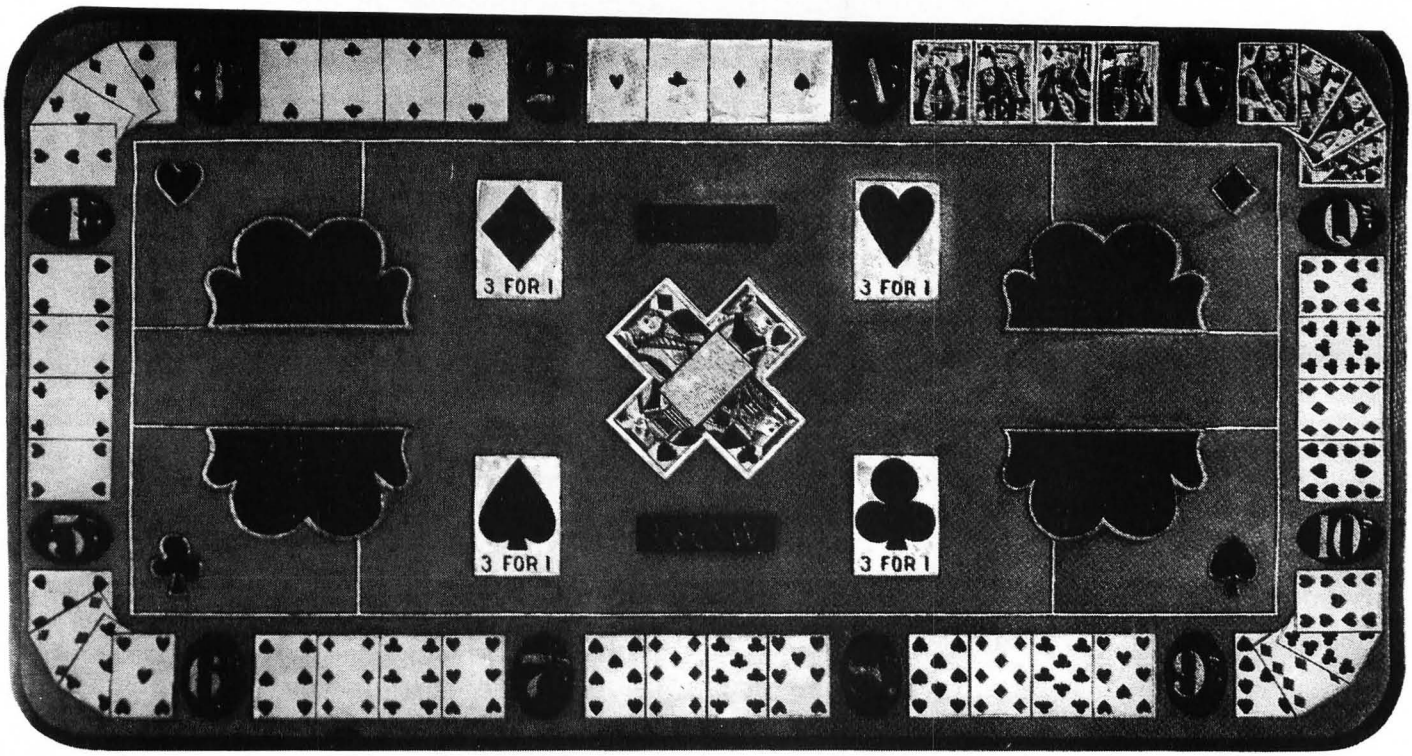
```
810 M$="":CLS:PRINT@262,"T h e   F o r t u n e   T e l l e r   S
a y s :":K=1:M=10*RND(5)+50
820 IFRND(5)=5THENM$="hold a sweepstakes.":K=2:GOTO890
830 IFRND(5)=5THENM$="hold a lottery.":K=3:GOTO890
840 IFRND(2)=2THENM$="collect from ":K=-1ELSEM$="pay to "
850 J=RND(N+1):IFJ<>IPTHEN860ELSEM$=M$+"everyone":M(IP)=M(IP)-K*
N*M:FORJ=1TON:M(J)=M(J)+K*M:NEXTJ:GOTO880
860 IFJ=N+1THENM$=M$+"the bank":M(IP)=M(IP)-K*M:GOTO880
870 M$=M$+N$(J):M(IP)=M(IP)-K*M:M(J)=M(J)+K*M
880 M$=M$+STR$(M)+" dollars."
890 QQ=0:D(0)=5000:D(1)=5040:D(2)=5020:D(3)=5007:PRINTCHR$(23):X
=USR(266)
900 M$=STRING$(32,32)+N$(IP)+", "+M$+STRING$(32,32):FORJ=1TOLEN(
M$)-31:PRINT@512,MID$(M$,J,31);:Q=USR(D(QQAND3)):QQ=QQ+1:FORTI=1
TO55:NEXTTI,J:M$=""
910 PRINT@966,"Touch any key to continue";:B$=INKEY$
920 IFINKEY$=""THEN920
930 IFK=2THEN460ELSEIFK=3THEN710ELSERETURN
```

High roller game. Instructions in line 940; all roll two dice in line 950. High roll determined in line 960 and 970. Single winner announced in line 980, tie roll causes reroll in lines 990 through 1010.

```
940 CLS:PRINT" H I G H   R O L L E R -- Everyone antes $20 and r
olls two dice. The highest total takes the $100 prize.  To roll
dice, touch":PRINT" any key on your turn.":GOSUB1470:PRINT@192,C
HR$(31)
950 FORJ=1TON:M(J)=M(J)-20:X=64+J*128:PRINT@X,N$(J):X=X+10:GOSUB
1450:D(J)=K:X=X+15:GOSUB1450:D(J)=D(J)+K:X=X-25:PRINT@X+35,"Your
 total is";D(J);:NEXTJ
960 HN=D(1):HM=1:FORJ=2TON:IFD(J)>HNTHENHN=D(J):HM=J:NEXTJELSENE
XTJ
970 K=0:FORJ=1TON:IFD(J)=HNTHENK=K+1:NEXTJELSENEXTJ
980 IFK=1THENPRINT@896,TAB(19+LEN(N$(HM))/2);N$(HM);" wins the p
ot!":M(HM)=M(HM)+100:B$=INKEY$:GOSUB1470:RETURN
990 PRINT@900,"We've got a tie!  Those high rollers will roll ag
ain!":B$=INKEY$:GOSUB1470:PRINT@192,CHR$(31);:X=192
1000 FORJ=1TON:IFD(J)<HNTHEND(J)=0:NEXTJELSEPRINT@X,N$(J):X=X+10
:GOSUB1450:D(J)=K:X=X+15:GOSUB1450:D(J)=D(J)+K:X=X-25:PRINT@X+35
,"Your total is";D(J);:X=X+128:NEXTJ
1010 GOTO960
```

Poker party subroutine. Instructions in line 1020. Three dice rolled in lines 1030 for all players. Check for three of a kind in line 1040, pair in line 1050, and straight in line 1070. Winner announced in line 1120. If a tie, start routine again.

```
1020 CLS:PRINT"P O K E R   P A R T Y --  Each player pays $20 an
d rolls three  dice.  The best poker hand (three of a kind > str
aight > pair)  wins $100.  Touch any key to roll dice.":GOSUB14
70:PRINT@192,CHR$(31):XX=256:FORI=1TON:M(I)=M(I)-20:NEXTI:HM=0
1030 FORI=1TON:PRINT@XX,N$(I):FORM=1TO3:J=I:X=XX+10*M:B$=INKEY$:
GOSUB1450:D(M)=K:NEXTM:P(I)=0
1040 IFD(1)=D(2)ANDD(1)=D(3)THENPRINT@XX+40,"Three of a kind!";:
P(I)=30+D(1):GOTO1100
1050 IFD(1)=D(2)THENP(I)=10+D(1)ELSEIFD(1)=D(3)THENP(I)=10+D(1)E
LSEIFD(2)=D(3)THENP(I)=10+D(2)
1060 IFP(I)>0THENPRINT@XX+40,"A Pair!";:GOTO1100
```




```
1070 FORJ=1TO3:IF(D(1)=D(2)+1ANDD(1)=D(3)+2)OR(D(1)=D(2)-1ANDD(1
)=D(3)-2)THENP(I)=20:NEXTJELSEHN=D(1):D(1)=D(2):D(2)=D(3):D(3)=H
N:NEXTJ
1080 HN=D(1):FORJ=2TO3:IFD(J)>HNTHENHN=D(J):NEXTJELSENEXTJ
1090 P(I)=P(I)+HN:IFP(I)>20THENPRINT@XX+40,"A Straight!";ELSEPRI
NT@XX+40,"Highest Roll is a";P(I);
1100 XX=XX+128:IFP(I)>HMTHENHM=P(I):NEXTIELSENEXTI
1110 K=0:FORI=1TON:IFP(I)=HMTHENK=K+1:J=I:NEXTIELSENEXTI
1120 IFK=1THENPRINT@914,N$(J);" wins the pot of $100":M(J)=M(J)+
100:GOSUB1470:RETURNELSEPRINT@905,"We have a tie ... so let's al
l play another hand!":GOSUB1470:PRINT@196,CHR$(31):XX=256:HM=0:G
OTO1030
```

Love thy neighbor routine. Set message in line 1150, adjust money in line 1160, and jump to display routine in line 890.

```
1130 CLS:PRINT@256,CHR$(23):K=0:M=RND(5)*10+50
1140 J=RND(N):IFJ=IPTHEN1140
1150 M$="show that you're a good neighbor and give "+N$(J)+STR$(
M)+" dollars."
1160 M(IP)=M(IP)-M:M(J)=M(J)+M:GOTO890
```

Easy come easy go routine. Give instructions and roll two dice in line 1170. Adjust money and return.

```
1170 CLS:PRINT"E A S Y   C O M E   E A S Y   G O  -- The bank w
ill pay you 10 times roll of 2 dice.  Touch any key to roll dice
, ";N$(IP);".":GOSUB1470:J=IP:X=279:GOSUB1450:M=K:X=289:GOSUB145
0:M=(M+K)*10:M(IP)=M(IP)+M:PRINT@470,"You win";M;"dollars."
1180 GOSUB1470:RETURN
```

Short message routines. Simple procedures to handle following special routines: Win a Few (1190), Unlucky Seven (1200), Lose a Few (1230), Even Steven (1240), Pot Luck (1300), Jackpot (1310), $100 Bonus (1320), Lose a Turn (1330), Tax Time (1340), Bonanza (1350), and Mad Money (1360).

```
1190 CLS:PRINT"W I N   A   F E W  -- The bank will pay you ten t
imes
the roll of one die.  Touch any key to roll die, ";N$(IP);".":GO
SUB1470:J=IP:X=284:GOSUB1450:M=K*10:M(IP)=M(IP)+M:PRINT@470,"You
 win";M;"dollars.":GOSUB1470:RETURN
1200 CLS:PRINT"U N L U C K Y   S E V E N  -- Roll two dice. If
the total is
seven you lose $100.  For any other total, you win $100.
Touch any key to roll dice, ";N$(IP);".":GOSUB1470:J=IP:X=279:GO
SUB1450:M=K:X=289:GOSUB1450:M=M+K
1210 IFM=7THENPRINT@470,"You lose 100 dollars!":M(IP)=M(IP)-100E
LSEPRINT@470,"You win 100 dollars!":M(IP)=M(IP)+100
1220 GOSUB1470:RETURN
1230 CLS:PRINT"L O S E   A   F E W  --  You must pay the bank te
n times the    roll of one die.  Touch any key to roll die, ";N$
(IP);".":GOSUB1470:J=IP:X=284:GOSUB1450:M=K*10:M(IP)=M(IP)-M:PRI
NT@470,"You lost";M;"dollars.":GOSUB1470:RETURN
1240 CLS:PRINT"E V E N   S T E V E N  --  You may bet up to $90
and roll two   dice.  If the total is even, you collect twice yo
ur bet.  Touch any key to roll dice.
";N$(IP);", how much will you bet ? ";:IFIP=NTHENB$=STR$(RND(4)+
5):GOTO1260
1250 B$=INKEY$:IFB$=""ORVAL(B$)<1THEN1250
1260 M=VAL(B$)*10:PRINTM;"dollars":GOSUB1470
1270 X=407:J=IP:GOSUB1450:P=K:X=417:GOSUB1450:P=P+K
1280 IFINT(P/2)*2=PTHENPRINT@598,"You win";2*M;"dollars.":M(IP)=
M(IP)+2*MELSEPRINT@598,"You lose";M;"dollars.":M(IP)=M(IP)-M
1290 GOSUB1470:RETURN
1300 CLS:PRINTCHR$(23):M=10*RND(5)+10:M(IP)=M(IP)-M:M$="why don'
t you sweeten the pot by giving the bank"+STR$(M)+" dollars.":K=
0:GOTO890
1310 CLS:PRINTCHR$(23):M=RND(5)*10+50:M$="you hit the jackpot! C
ollect"+STR$(M)+" dollars.":K=0:M(IP)=M(IP)+M:GOTO890
1320 CLS:PRINTCHR$(23):M(IP)=M(IP)+100:M$="please accept this $1
00 bonus!":K=0:GOTO890
1330 CLS:PRINTCHR$(23):M$="you just lost this turn!":K=0:GOTO890
1340 CLS:PRINTCHR$(23):M$="pay $100 tax to the bank!":M(IP)=M(IP
)-100:K=0:GOTO890
1350 CLS:PRINTCHR$(23):M=RND(5)*10:M(IP)=M(IP)+M*N:FORJ=1TON:M(J
)=M(J)-M:NEXTJ:M$="WHAT A BONANZA! EVERYONE PAYS YOU"+STR$(M)+"
DOLLARS.":K=0:GOTO890
```

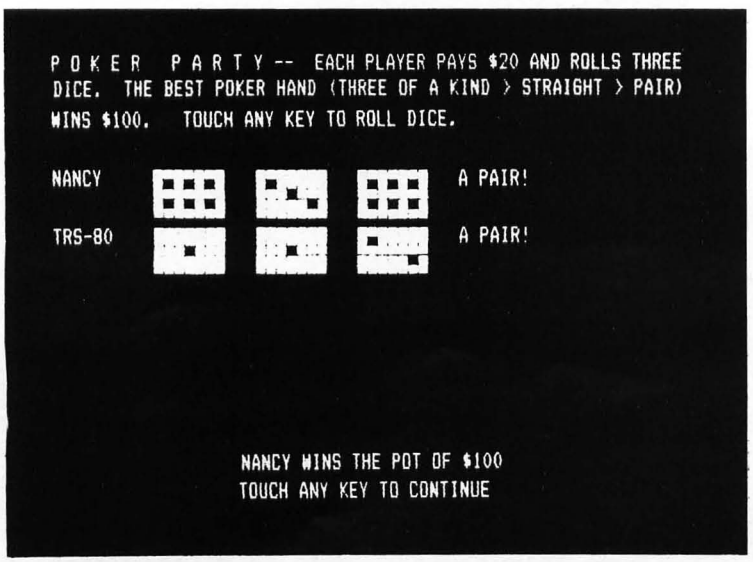
```
1360 CLS:PRINTCHR$(23):M=RND(5)*10:M(IP)=M(IP)-M*N:FORJ=1TON:M(J
)=M(J)+M:NEXTJ:M$="this ought to make you mad.  Give everybody"+
STR$(M)+" dollars.":K=0:GOTO890
```

Game instructions. Called optionally from initialization section.

```
1370 CLS:PRINTCHR$(23)"This is the game of GAMBLER!";PRINT:PRINT
"You and the TRS-80 are in a con-test to see who will build his
 $100 bankroll into $1000 first. Money is made and lost through
aseries of games of chance --
from horse racing and dice games";
1380 PRINT"to lotteries and sweepstakes! Ifyou should lose all y
our money
your IOU will be accepted ( as
long as you pay it back with
interest ). Good luck!":K=0:GOTO910
```

Utility subroutines. Lines 1390 through 1440 set up the sound-making routine in SOUND$. Lines 1450 through 1460 wait for input before stopping on one of the six dice pictures at print position X. The computer's dice choice is made in line 1450. Lines 1470 and 1480 print a message and wait for keyboard response.

```
1390 SOUND$="////////////////////////////": '28 SLASHES
1400 I=VARPTR(SOUND$):J!=PEEK(I+1)+256*PEEK(I+2):J=J!+65535*(J!>
32767)
1410 FORK=1TO36:READA$:NEXTK:FORK=JTOJ+26:READX:POKEK,X:NEXTK
1420 IFPEEK(16396)=201THENPOKE16526,PEEK(I+1):POKE16527,PEEK(I+2
)ELSECMD"T":DEFUSR0=J:POKE14308,0
1430 RETURN
1440 DATA 205,127,10,77,68,62,1,105,211,255,45,32,253,60,
         105,211,255,45,32,253,13,16,238,175,211,255,201
1450 B$=INKEY$:IFJ=NTHENFORL=1TORND(50):K=RND(6):PRINT@X,D$(K);:
Q=USR(1E4+RND(99)):NEXTL:RETURN
1460 K=RND(6):PRINT@X,D$(K);:Q=USR(1E4+RND(99)):IFINKEY$=""THEN1
460ELSERETURN
1470 PRINT@978,"Touch any key to continue";
1480 IFINKEY$<>""THENPRINT@978,CHR$(30);:RETURNELSEIFRND(9)=1THE
NQ=USR(1E4+RND(99)):GOTO1480ELSE1480
```



NUCLEAR SUBMARINE ADVENTURE

TRS-80® DV
APPLE-DV

by Steven Neighorn

Nuclear Submarine Adventure **is an adventure for an Apple (with Applesoft) or TRS-80®, requiring 32K RAM and disk drive. It is included as a bonus program on this month's Apple and TRS-80® Disk Versions.**

Due to the growing threat of Communist expansion and nuclear proliferation, the U.S.S. Nautilus has been totally refitted with modern equipment. This includes 16 missile tubes carrying the new Trident-1 nuclear missiles; a water-cooled reactor; two torpedo tubes armed with MK-48 torpedoes; and totally new submarine-quieting, mobility, and self-defense systems. For the crew there is a new and separate health room, and an easy-access passageway in the fore and aft sections of the ship.

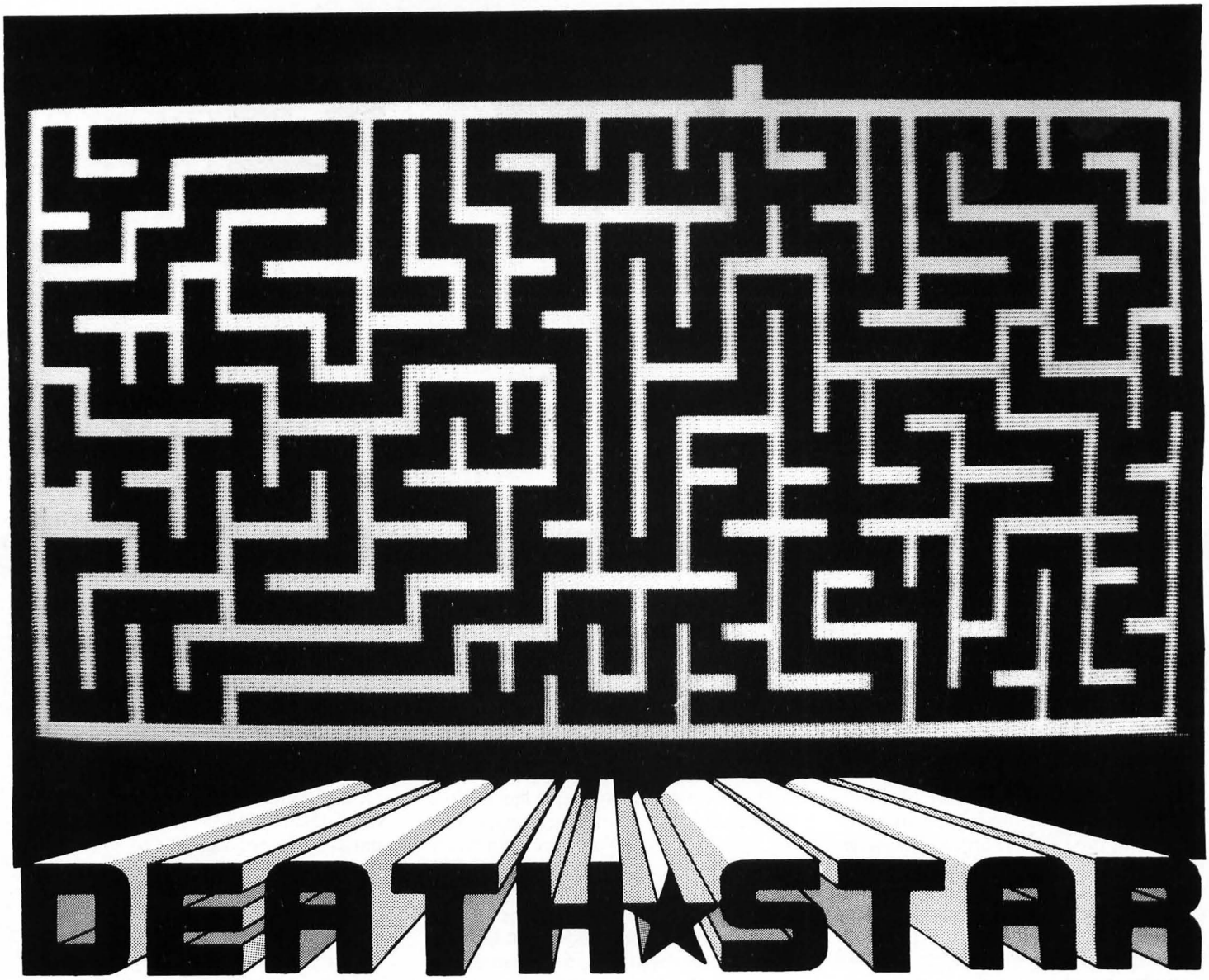
You are a new crew member aboard the Nautilus. Although you have received extensive training before starting your tour of duty, experience is the best teacher. You will learn much during your stay. You are currently aboard with a skeleton crew, testing to see if (and how) the ship can be safely operated in the event that most of the crew should become incapacitated.

Remember that you, as a member of the United States Submarine Corps, are helping to ensure that the United States continues to maintain a credible, surviving deterrent to nuclear war in the 21st century.

NOTES ON PLAYING

This adventure recognizes 22 commands and 45 objects, and has 41 locations. Only the first three letters of any word are needed for the computer to understand the meaning of the command. Some useful commands are: INVENTORY, to list what you are currently carrying; LOOK, to look at an object; and QUIT, to end the game. There is no HELP command, but you can use SAVE and LOAD to preserve and recall a game in progress.

You can carry up to five objects at one time. Likewise, only five objects can be at any single location at one time. In some rooms where there are a lot of objects, you might not be able to read certain objects, because of the way the display scrolls. All you have to do is to move to another room where there are not as many objects.



ATARI® DV

by Matt Rutter

***Death Star* is a graphics game program for a 32K ATARI® with disk drive. It is included as a bonus program on this month's ATARI® Disk Version.**

You are a spy for the rebel alliance and you have just been transported into the nuclear reactor room of the infamous *Death Star*. Your mission is to find and activate the faulty reactor, which will quickly set off a chain reaction capable of destroying the *Death Star*; and then to escape with your own life.

The computer will draw a map of the reactor room (different each time you play), with you at the left side, the reactor at the top, and the exit on the right side. You will then be shown a three-dimensional view of the maze from where you are standing, facing east (right). The only items you are carrying are a compass and an electronic map with failing energy cells. The compass direction is displayed at the bottom of the screen. You can use the map at any time simply by pressing "M"; however, your energy cells will hold out only for eight viewings of the map.

To move through the corridors you use the arrow keys as follows:

Up-arrow: Move one space forward.
Left-arrow: Turn left.
Right-arrow: Turn right.
Down-arrow: Turn about-face.

You must use these keys to move through the twisty maze of passages until you arrive at the switch that activates the nuclear reactor. When you face this switch, you will be able to turn it on by pressing "P". After doing this, you have only a short time to find the exit and escape before the countdown reaches zero and the reactor explodes — taking you with it. (If you find yourself needing either less or more challenge, you can adjust the initial countdown value, CD, in line 1140.)

Good luck, and may the Force be with you.


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.

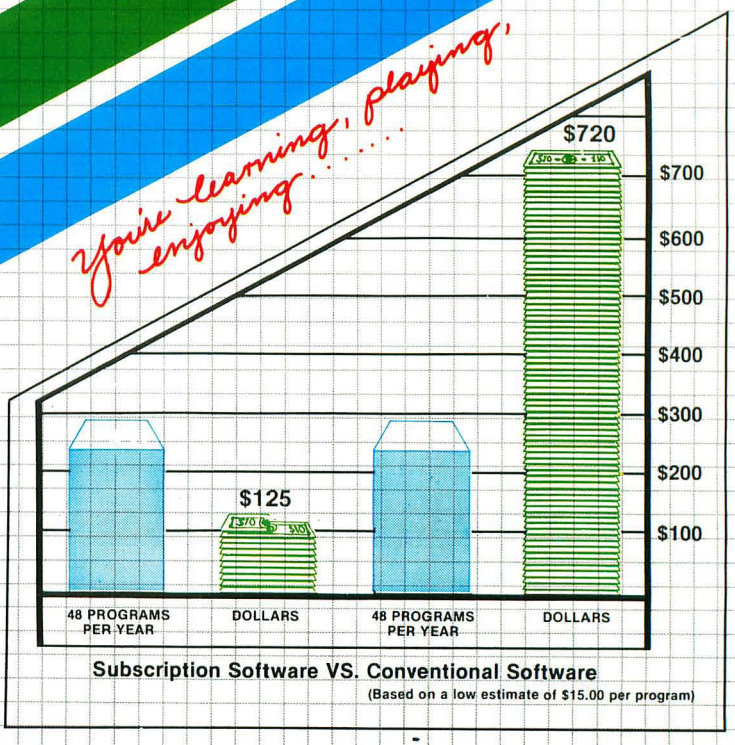
A subscription to **SoftSide** DV is more than a savings — it's an investment in an ongoing library of software — multiple and Machine Language programs which would be difficult to read and even more difficult to key into your system. You'll spend your software dollars wisely with a subscription. Where 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



Subscription Software VS. Conventional Software
(Based on a low estimate of $15.00 per program)

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

OOPS! Typo!

DATA TIME

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 _____

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.

Smasher Typos!

Stop Typing....


Stop
typing
line listings...
Start
typing
commands!

K-Byter ATARI®

Letter Writer

An ATARI® K-Byter by Harry Caporuscio, Dhahran, Saudi Arabia

This is a simple word processor program which allows you to type a page of text on your ATARI® and dump the screen to a printer. You simply input your text line by line, pressing RETURN at the end of each line. Editing is accomplished by using the control arrow keys. When you have entered 23 lines of text, they will automatically be sent to the printer just as they appear on the screen. If you want to dump the screen prior to filling all 23 lines, type an asterisk (*).

While the program is written for a 40-column printer, it could be adapted to 80 columns by using a concatenation technique. Another possible modification would be a routine to save the text to disk for later recall or processing.

```
1 REM LETTER WRITER
2 REM BY HARRY CAPORUSCIO
3 REM AUGUST 1981
5 POKE 82,0
10 DIM L$(39)
20 ? CHR$(125)
30 OPEN #3,4,0,"S:"
40 FOR I=0 TO 22:INPUT L$
50 IF L$="*" THEN 110
60 NEXT I
110 FOR DOWN=0 TO 22
120 POSITION 1,DOWN
130 FOR ACROSS=1 TO 39
140 GET #3,L
150 L$(ACROSS,ACROSS)=CHR$(L)
160 NEXT ACROSS
170 IF L$(1,1)="*" THEN END
180 LPRINT L$(1,39)
190 NEXT DOWN
200 RUN
```

K-Byter Apple

Letter Spitter

An Apple K-Byter by Jon Voskuil, *SoftSide*

This is purely a fun program, written with young children in mind but amusing for oldsters as well. A creature of some sort appears at the bottom of a blank screen, and waits for you to press a key. The character you type is spit by the creature up onto the screen, with an appropriate sound. As you type more characters, they too are spit upward and take their place in line, while the creature moves around at random at the bottom. When nine rows have been filled, the creature starts over at the first position once more — but without clearing the screen, so that some letters on lower rows may be knocked out of place by the new letter as it zips past.

Pressing the RETURN key will move you to the next line down, as usual; and pressing the ESC key will clear the screen. Because of the ONERR statement in line 100, no key except RESET will cause the program to terminate.

```
1  REM --------------------
2  REM   "LETTER SPITTER"
3  REM    BY JON VOSKUIL
4  REM --------------------
100  ONERR  GOTO 130
110 S =  - 16336
120 G$ = "/Y" +  CHR$ (92)
130  HOME
140 INCR = 6
150 P5 = .5
160 C = 1:R = 1
170 COL =  INT ( RND (1) * 38) +
    2
180 ROW = 20 +  INT ( RND (1) * 5
    )
190 VO = ROW:HO = COL
200 VI = (ROW - R) / INCR
210 HI = (COL - C) / INCR
220  VTAB ROW - 1: HTAB COL - 1: PRINT
    G$;
230  IF  PEEK ( - 16384) < 127 THEN
    230
240  GET X$: POKE  - 16368,0
250  POKE S, PEEK (S): POKE S, PEEK
    (S): POKE S, PEEK (S): POKE
    S, PEEK (S): POKE S, PEEK (S
    ): POKE S, PEEK (S)
260  IF X$ =  CHR$ (27) THEN 130
270  IF X$ =  CHR$ (13) THEN C =
     - 1:R = R + 2: GOTO 350
280  FOR I = 0 TO INCR
290 V = P5 + ROW - VI * I: VTAB V
300 H = P5 + COL - HI * I: HTAB H
310  PRINT X$;
320  VTAB VO: HTAB HO: PRINT " ";
330 VO = V:HO = H
340  NEXT
350  POKE S, PEEK (S): POKE S, PEEK
    (S)
360 C = C + 2: IF C > 38 THEN C =
    1:R = R + 2
370  IF R > 18 THEN R = 1
380  VTAB ROW - 1: HTAB COL - 1: PRINT
    "  ";
390  GOTO 170
```

K-Byter TRS-80®

Row Switch

A TRS-80® K-Byter by Carl Bevington, Salem, OH

This is a game which is played on a board five squares across and two squares deep. Five markers are initially placed in the five upper positions of the board. The object is to move these markers so that they occupy the five lower positions on the playing board.

The markers can move only according to the following rules: (1) The marker on the extreme right may move up or down without restriction, while (2) The first four markers may move up or down only when the marker to their immediate right is in the up position and all other markers to the right are in the down position.

A minimum of 21 moves is necessary to complete the exchange of markers and end the game.

```
100 DEFINTA-Z:DIMP(80),C(80):CLS:FORX=2TO125:SET(X,3):SET(X,4):S
ET(X,15):SET(X,16):SET(X,27):SET(X,28):NEXTX:FORX=2TO122STEP24:F
ORY=3TO28:SET(X,Y):SET(X+1,Y):SET(X+2,Y):SET(X+3,Y):NEXTY,X
110 A$=CHR$(191)+CHR$(191)+CHR$(191)+CHR$(191)+CHR$(191):FORI=65
TO74:READP(I):C(I)=0:PRINT@P(I),CHR$(I);:NEXTI:FORI=65TO69:C(I)=
I:PRINT@P(I)+66,A$;:PRINT@P(I)+322,"    ";:NEXTI:RESTORE:Z=1:DA
TA131,143,155,167,179,387,399,411,423,435
120 PRINT@665,"MOVE NUMBER";:PRINT@677,Z;
130 PRINT@786,"                                                ";:PR
INT@786,"MOVE FROM?";
140 F$=INKEY$:IFF$=""THEN140ELSE150
150 PRINT@798,F$;:PRINT@803,"MOVE TO?";
160 T$=INKEY$:IFT$=""THEN160ELSE170
170 PRINT@813,T$;:F=ASC(F$):T=ASC(T$):IFF>74ORT>74THEN250
180 IFC(F)=0THEN250
190 IFC(T)<>0THEN250
200 IFF$T=4550ANDC(66)=66ANDC(67)+C(68)+C(69)=0THEN260
210 IFF$T=4686ANDC(67)=67ANDC(68)+C(69)=0THEN260
220 IFF$T=4824ANDC(68)=68ANDC(69)=0THEN260
230 IFF$T=4964ANDC(69)=69THEN260
240 IFF$T=5106THEN260
250 PRINT@786,"MOVE NOT POSSIBLE, TRY AGAIN.";:FORQ=1TO700:NEXTQ
:GOTO130
260 C(F)=0:C(T)=T:PRINT@P(T)+66,A$;:PRINT@P(F)+66,"       ";:Z=Z
+1
270 IFC(65)$C(66)$C(67)$C(68)$C(69)=0ANDC(70)$C(71)$C(72)$C(73)$
C(74)<>0THEN280ELSE120
280 PRINT@784,"  YOU FINISHED IN ";Z-1;" MOVES. ";:FORQ=1TO700:N
EXTQ:GOTO110
```

Modify EDTASM For The Model III

by Randy Hawkins

First in a two-part series.

How many of you who traded in your Model I for a Model III had the following happen to you — After playing with your brand-new Model III computer for a while, you decide to get down to serious business and begin programming using the *Editor Assembler* program. Setting the cassette on the low 500 baud rate, you load the Model I *Editor Assembler* program with no trouble. But when you try to execute it, you are baffled — that is, unless you graduated from Tokyo High School. Not only is the title filled with Japanese characters, but none of the standard editor assembler functions works properly. Your $30 investment has gone down the drain, and you are without one of your most valuable programming tools for who knows how long!

Not only did this happen to me, but it made me very angry. I was not about to lose a perfectly good program. I was convinced that with a little detective work I could find a way to salvage my program . . . and I did! This article will describe a way to fix the program and allow it to operate just as it did on the Model I using a simple BASIC program to make the changes. In addition, once you have your Model III *Editor Assembler* (hereafter referred to by its file name, *EDTASM*) up and running, you can use it with a simple Machine Language program which will be presented next month that will translate 500 baud SYSTEM tapes into 1500 baud tapes.

The information presented in this article assumes you have the *EDTASM* program version 1.2 and a standard 16K or larger cassette-based Model III. If you now own a Model III but do not own the *EDTASM* program, you could conceivably purchase the program and make the changes in this article to use it on the Model III. However, it might be wise to wait and see how this revised version compares with the one soon to be released by Radio Shack. On the other hand, if you kept your copy of *EDTASM* when you sold your Model I and purchased a Model III, then you cannot go wrong by following the instructions below.

How to Fix it

The first step in our repair work is the BASIC program listing found with this article. Type in this program as listed. Granted, the scores of DATA statements are very tedious. But without the convenience of an editor assembler program, it is our only way to change the program at this time. Since the program is almost entirely numerical entry, a method of checksums is used to help alert you to incorrect entries.

Once you have keyed in the program, CSAVE it before running it. This is to prevent you from losing your hours of hard work should the program accidentally return you the Cass? question.

With a copy on tape, now RUN the program. After POKEing the correction instructions into high memory, the program will prompt you to load the original version of the editor assembler. Type in the word "EDTASM" and press enter. (The baud rate has correctly been set at 500 for you.)

Once the program has loaded, enter a slash and "25000". Note: That is not just a slash, but a slash and the number "25000". In a few seconds, the program will instruct you to load a blank tape into the recorder. Do so, set it on record, and touch any key. A new version of the *EDTASM* modified for the Model III will be saved on your tape. If you want additional copies, leave the cassette on record, and touch another key when instructed to do so. The program will continue to record as many copies as you desire since it is caught in an endless loop. To escape, press the RESET button.

How to Use it

The revised version of the editor assembler saved to your tape is now ready to use. Load it using the SYSTEM command, file name EDTASM. Since the program loads at 1500 baud, it will now load in under a minute rather than several minutes as it did on the Model I.

```
10 CLS :PRINT :PRINT "This program will create a new, revised ve
rsion of the S-80     EDITOR/ASSEMBLER for the Model III. The ch
anges included assume you are using version 1.2 of EDTASM. The r
outine is now being   loaded . . ."
20 SA=24999 :LN=100
30 FOR J=1 TO 36 :CS=0 :FOR I=1 TO 10 :READ X(I) :CS=CS+X(I) :NE
XT I :READ XX :IF XX<>CS THEN PRINT "Checksum error in line";LN
:END
40 FOR I=1 TO 10 :POKE SA+I,X(I) :NEXT I :SA=SA+10 :LN=LN+10 :NE
XT J
50 PRINT :PRINT "The special routine is now ready. At the '*?' p
rompt below," :PRINT "load the original EDTASM program in your c
assette and set it" :PRINT "to PLAY. Type in 'EDTASM' and press
<ENTER>."
60 PRINT "When the program has loaded successfully, type in" :PR
INT TAB(22) "*? /25000" :PRINT "as shown above. Follow the direc
tions to save the revised" :PRINT "EDITOR/ASSEMBLER for the Mode
l III to your own tape."
70 POKE 16913,0 :SYSTEM
100 DATA 33,239,67,17,173,98,6,3,205,215,1056
110 DATA 9,33,96,68,17,176,98,6,3,205,711
120 DATA 215,9,33,170,69,17,179,98,6,10,806
130 DATA 205,215,9,33,201,72,17,189,98,6,1045
140 DATA 6,205,215,9,62,176,50,227,72,62,1084
150 DATA 95,50,14,67,33,248,1,34,222,70,834
160 DATA 33,100,2,34,67,71,34,70,77,34,522
170 DATA 58,79,34,64,79,34,73,79,34,173,709
180 DATA 89,34,179,89,34,183,89,34,205,92,1028
```

```
190 DATA 34,211,92,33,150,2,34,88,77,175,896
200 DATA 33,166,77,119,35,119,35,119,33,135,871
210 DATA 2,34,53,79,33,53,2,34,91,77,458
220 DATA 34,112,77,34,121,77,34,150,77,34,750
230 DATA 180,77,34,187,77,62,1,50,17,66,751
240 DATA 33,201,98,205,167,40,205,73,0,205,1227
250 DATA 135,2,33,195,98,62,85,205,100,2,917
260 DATA 6,6,126,205,100,2,35,16,249,17,762
270 DATA 0,93,33,0,67,62,60,205,100,2,622
280 DATA 6,64,120,205,100,2,14,0,125,205,841
290 DATA 165,98,124,205,165,98,126,35,205,165,1386
300 DATA 98,229,55,63,237,82,225,40,8,16,1053
310 DATA 241,121,205,100,2,24,214,16,2,24,949
320 DATA 7,62,0,205,165,98,16,251,121,205,1130
330 DATA 100,2,62,120,205,100,2,33,173,69,866
340 DATA 125,205,100,2,124,205,100,2,205,248,1316
350 DATA 1,24,138,245,129,79,241,205,100,2,1164
360 DATA 201,195,36,48,195,115,4,195,194,3,1186
370 DATA 175,50,17,66,195,138,70,77,79,68,935
380 DATA 69,76,51,69,68,84,65,83,77,10,652
390 DATA 10,80,82,69,80,65,82,69,32,84,653
400 DATA 79,32,82,69,67,79,82,68,32,82,672
410 DATA 69,86,73,83,69,68,32,69,68,84,701
420 DATA 65,83,77,10,84,72,69,78,32,80,650
430 DATA 82,69,83,83,32,65,78,89,32,75,688
440 DATA 69,89,32,84,79,32,66,69,71,73,664
450 DATA 78,46,0,0,0,0,0,0,0,0,124
```

Bugs, Worms

and other undesirables

There are two problems with the Apple program, 'Music Machine,' published in the November, 1981, issue. One is that the mod function subroutine does not work quite properly because of a rounding error inherent in Applesoft. This can be easily corrected by modifying line 60 as follows:

```
60 R =  INT ((NUM / MOD -  INT (N
    UM / MOD)) * MOD + .00001): RETURN
```

The second problem involves processing rests. Thanks to Cary Bradley for the following modifications:

```
50  IF NT = 0 THEN  FOR I = 1 TO
    2 * (D - (D * TEMPO - D) / 8
    ): NEXT I: RETURN
55  & TNT,D - (D * TEMPO - D) / 8
    : RETURN
```

```
2410 NN = 0: IF C$ = "R" THEN 259
    0
```

TRS-80® One Liner

```
1 A$=INKEY$:IFF=0CLS:CLEAR999:PRINTCHR$(95);:F=1:GOTO1ELSEIFA$="
"THEN1ELSEIFA$=CHR$(13)ORC=61THENLPRINTB$:B$="":C=0:PRINTCHR$(29
)+CHR$(26)+CHR$(95);:GOTO1ELSEPRINTCHR$(24)+A$+CHR$(95);:IFA$=CH
R$(8)B$=LEFT$(B$,LEN(B$)-1):C=C-1:GOTO1ELSEB$=B$+A$:C=C+1:GOTO1
```


WOODY POPE
1029 MISTY WAY
GARLAND, TX, 75040

COMMANDing BASIC

by Alan J. Zett

The USR subroutine presented in this article is an enhancement to TRS-80® Level II and Disk BASIC requiring at least 16K RAM.

This column is basically a concept with an accompanying program, designed to enhance the operation of BASIC with Machine Language. This can be accomplished by adding new commands, such as the SOUND command from the November issue, or by modifying the commands already available to be more flexible. The latter will be the subject of the next two installments. The subject for this month is "RESTOREing to a line number;" but before I get too far into this month's column, I wish to expand a little more on SOUND.

ERRATA: The documentation for SOUND stated that the command would stay in memory "until a system reset occurs or until any statement that clears variables is executed." This is not so. SOUND is not effected by a statement that clears variables. Because the length of the BASIC program that pokes in SOUND is a bit long to be used within a BASIC program, at the last minute I decided to change the way in which it would be stored in memory. It now resets memory size by itself and is self-relocating. Unfortunately, the documentation had already been printed up. (This month's routine can, however, be wiped out with a CLEAR statement because it is only a USR routine poked into a variable.)

The best way to use SOUND (or any of my new commands), is to run the initializing routine (the BASIC program) and then erase the program by typing NEW. SOUND will still be in memory so you can then load in a program containing SOUND statements and it will work properly. The initializing program was also written to be included as a subroutine for an all-in-one, self-contained program using SOUND.

Also, because BASIC handles an IF statement differently than the other commands, it is necessary to precede SOUND with a colon if used after a THEN or ELSE statement, otherwise a ?SN error will occur. For example: IF A=1 THEN SOUND 11,11 ELSE SOUND 200,1000 will not work, but IF A=1 THEN :SOUND 11,11 ELSE :SOUND 200,1000 will. This is true of all new commands that will be appearing in this column. The Machine Code program itself could be modified to handle this special case, but since it is already quite long, it's easier just to use the colon.

Another problem with tying into the error relay is that ROM will have already cleared the line below the current cursor position to make way for an error message. This means that if you type PRINT@ 966,"HI";:SOUND 11,11 the entire screen will scroll one line. To get around this I use a PRINT@0; before the start of any sound. Note that the sound statement in a FOR. . .NEXT loop will not generate multiple line feeds, so a PRINT@0; before a set of SOUND statements will work fine. For an example of a program using the SOUND statement, see *Space Rescue* in an upcoming issue.

And now: On to this month! One of the better features of Atari BASIC is the ability to RESTORE the data pointer to a specified line number. TRS-80® BASIC already has this ability but lacks the coding to make it work.

The short USR routine presented here will allow a line number to be specified as the start of the data pointer. BASIC will react to this number in many different ways. For example: A normal RESTORE will set the data pointer to the first program line. When a READ statement is processed it will start searching each line from the start of the data pointer forward for the first DATA keyword it can find. Note that data need not be on the first line pointed to by the data pointer.

With the USR routine, a line number is searched for and, if found, is marked as the starting point in the data pointer. If there are no data past that point, an ?OD or Out of Data error will occur because the program is trying to read data that are not there. If a RESTORE to line 150 is executed and line 150 exists, then if there are data at lines 100 and 200, line 200 will be used for data instead of line 100.

The format for the USR call is:

X=USR(line number)

where "line number" is any number between 0 and 32767. The line number may also be negative for numbers higher than 32767. The reason for this is that any USR argument must be an integer or an ?OV error will occur.

Program 1 is a listing of a BASIC program that will POKE in the RESTORE routine, followed by Program 2, a demonstration program. After that is Program 3, an unassembled listing of the source code for all you Machine Code hackers out there.

Program 1

```
1 CLS:CLEAR:REST$="THIS IS WHERE IT GOES!":RESTORE
2 DATA205,127,10,229,209,205,44,27,48,6,11,237,67,255,64
3 DATA201,30,14,225,195,166,65:Z=VARPTR(RE$):Y=PEEK(Z+1)+PEEK(Z+
2)*256:FORX=YTOY+21:READV:POKEX,V:NEXT:IFPEEK(16396)=201THENPOKE
16526,PEEK(Z+1):POKE16527,PEEK(Z+2)ELSECMD"T":POKE14308,0:DEFUSR
=Y+65536*(Y>32767)
```

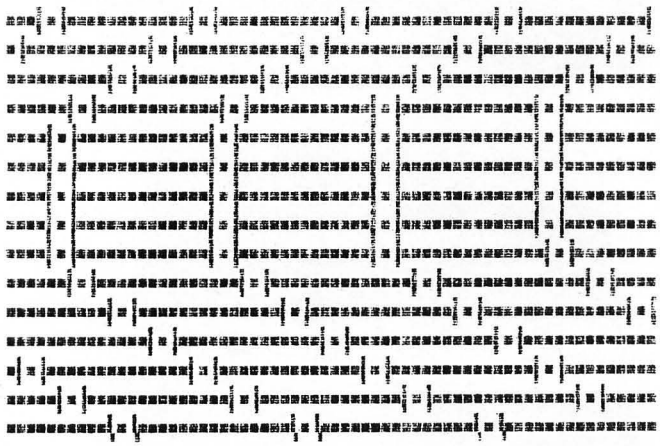
Program 2

```
1 CLS:CLEAR:REST$="THIS IS WHERE IT GOES!":RESTORE
2 DATA205,127,10,229,209,205,44,27,48,6,11,237,67,255,64
3 DATA201,30,14,225,195,166,65:Z=VARPTR(RE$):Y=PEEK(Z+1)+PEEK(Z+
2)*256:FORX=YTOY+21:READV:POKEX,V:NEXT:IFPEEK(16396)=201THENPOKE
16526,PEEK(Z+1):POKE16527,PEEK(Z+2)ELSECMD"T":POKE14308,0:DEFUSR
=Y+65536*(Y>32767)
10 A=70: CLS
20 X=USR(20): READ A$: PRINT A$
30 X=USR(A): READ A$: PRINT A$
40 X=USR(A+10): READ A$: PRINT A$
50 X=USR(A*3-100): READ A$: PRINT A$: END
60 DATA "LINE 60 TEST DATA"
70 DATA "LINE 70 TEST DATA"
80 GOTO 80
90 DATA "LINE 90 TEST DATA"
100 GOTO 100
110 DATA "LINE 110 TEST DATA"
120 GOTO 120
```

Program 3

```
00100          ORG     0F000H  ;BASIC PROGRAM IS RELOCATABLE
00110 FLINE    EQU     1B2CH   ;FIND A LINE NUMBER IN MEM.
00120 RSTPTR   EQU     40FFH   ;RESTORE DATA POINTER
00130 GETUSR   EQU     0A7FH   ;GET USR ARG. INTO HL
00140 ERELAY   EQU     41A6H   ;ERROR RELAY
00150 START    CALL    GETUSR  ;GET USR ARG.
00160          PUSH    HL      ;TRANSFER TO
00170          POP     DE      ; DE REGISTER.
00180          CALL    FLINE   ;SEARCH FOR LINE NUMBER
00190          JR      NC,UL   ;IF LINE DOESN'T EXIST
00200 RESTOR   DEC     BC      ;SUBTRACT ONE
00210          LD      (RSTPTR),BC     ;SAVE NEW POINTER
00220          RET             ;RETURN TO BASIC
00230 UL       LD      E,14    ;MAKE A ?UL ERROR
00240          POP     HL      ;REMOVE USR RET ADDRESS
00250          JP      ERELAY  ; AND PRINT IT.
00260          END
```

TRS-80 One Liners



```
1 CLS:CLEAR400:A$=CHR$(149)+CHR$(140)+CHR$(170):FORJ=1TO150:FORI
=1TO20:PRINTA$;:NEXT:A$=A$+CHR$(140):NEXT
```

Mike Kenekes
Harrisville, PA

```
1 A$=INKEY$:IFF=0CLS:CLEAR999:PRINTCHR$(95);:F=1:GOTO1ELSEIFA$="
"THEN1ELSEIFA$=CHR$(13)ORC=61THENLPRINTB$:B$="":C=0:PRINTCHR$(29
)+CHR$(26)+CHR$(95);:GOTO1ELSEPRINTCHR$(24)+A$+CHR$(95);:IFA$=CH
R$(8)B$=LEFT$(B$,LEN(B$)-1):C=C-1:GOTO1ELSEB$=B$+A$:C=C+1:GOTO1
```

Woody Pope
Garland, TX

```
1 X$=CHR$(191):FORX=0TO1E33STEP.08:PRINT@961,"-1";:PRINT@1021,"+
1";:PRINT@992,"0";:PRINT@992+(SIN(X)*29),X$;:PRINT@992+(SIN(X*1.
5)*29),X$;:PRINT@992+(SIN(X*2)*29),X$:NEXT
```

Alan J. Zett
SoftSide

TRS-MAN

by George Delp

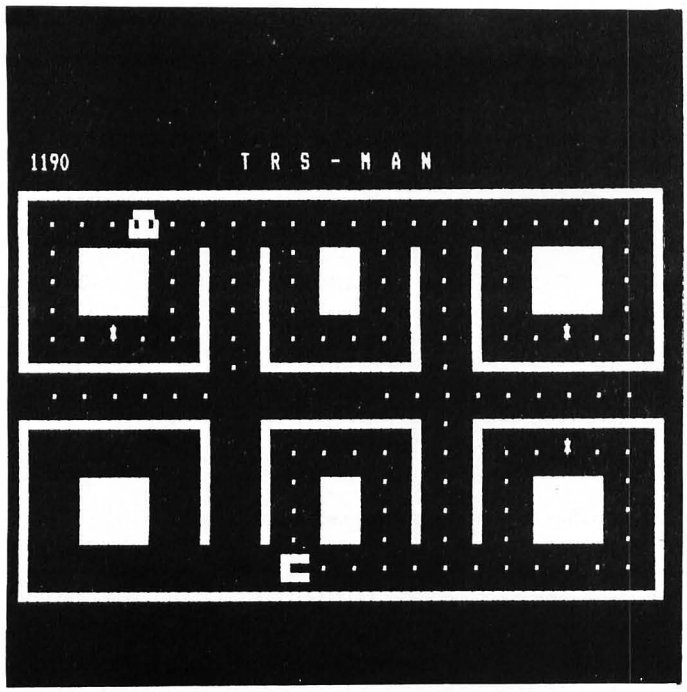
***TRS-Man* is an arcade game for the TRS-80® Model I or III and requires 16K of RAM.**

This is a real-time game very similar to a popular arcade game. You control a little mouth with the four arrow keys, and try to "eat" all of the little white dots (periods) on the screen. Each white dot is worth ten points. There are one to four computer-controlled creatures chasing you. When you run into one of the creatures or it runs into you, it counts as a hit against you. After you've been hit three times, the game ends.

There are also four stars (asterisks) on the screen. When you eat one of them, you get 100 points and all of the squares on the board light up. While the squares are lit, you can eat the computer-controlled creatures. They are worth 200-800 points. After awhile, the squares will turn dark, and the computer-controlled creatures will start chasing you again.

Variables

A, B: FOR-NEXT loops and miscellaneous.
AD: The difference between your position and that of the creatures chasing you.
CO: Counter for how long the squares stay lit.
D: Equals 15360 (start of video RAM).
DI: Direction of the player's mouth (1 = up, 2 = down, 3 = left, 4 = right).
DI(1-4): Direction of the creatures chasing you.
D1$-D8$: Graphics strings used in printing the playing field.
F1$-F2$: Graphics strings used in lighting the squares after you eat a star.
FL: Flag which is set to 1 when you can eat the creatures chasing you.
I, J: FOR-NEXT loops and miscellaneous.
NF: Number of dots and stars that you have eaten.
NO: Number of creatures out on the playing field chasing you around.
PC$: Graphics string used in printing the creatures chasing you.
PK: ASCII value of what previously occupied the player's position.
PK(1-4): ASCII value of what previously occupied the creature's position.
PL$(1-4,1-2): Graphics strings used in printing the player's mouth.
PO: Position of the player's mouth.
PO(1-4): Position of the creatures chasing you.
SC: The player's score.
T, X, XX: FOR-NEXT loops and miscellaneous.
Z: Equals PEEK(14400); used to scan keyboard for arrow keys.




```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         TRS-80  BASIC         $
$           "TRS-MAN"           $
$      Author: George Delp      $
$      (c) 1982  SoftSide      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialize variables, clear string space, and print the title page.

```
10 CLEAR500:DEFINTA-Z:DIMPL(4,2):D=15360
20 PC$=CHR$(182)+CHR$(191)+CHR$(185):PL$(1,1)=CHR$(191)+CHR$(176
)+CHR$(191):PL$(2,1)=CHR$(191)+CHR$(131)+CHR$(191):PL$(3,1)=CHR$
(179)+CHR$(179)+CHR$(191):PL$(4,1)=CHR$(191)+CHR$(179)+CHR$(179)
21 CLS:PRINT@5,PC$PC$PC$PC$PC$PC$"   "PC$PC$PC$PC$"          "PC$PC$
PC$:PRINT@75,PC$"          "PC$"          "PC$"   "PC$"          "P
C$:PRINT@139,PC$"          "PC$"          "PC$"   "PC$
22 PRINT@203,PC$"          "PC$PC$PC$PC$"          "PC$PC$PC$:PRIN
T@267,PC$"          "PC$"   "PC$"                    "PC$:PRINT@
331,PC$"          "PC$"      "PC$"      "PC$"          "PC$
23 PRINT@395,PC$"          "PC$"          "PC$"      "PC$PC$PC$:PR
INT@450,"";:FORT=1TO20:PRINT".  ";:NEXT
24 PRINT@517,PC$"          "PC$"          "PC$"          "PC$"
    "PC$:PRINT@581,PC$PC$"   "PC$PC$"      "PC$"   "PC$"      "P
C$PC$"      "PC$:PRINT@645,PC$"   "PC$"   "PC$"   "PC$"
"PC$"   "PC$"   "PC$"   "PC$
25 PRINT@709,PC$"   "PC$"   "PC$"   "PC$"          "PC$"   "PC$"
  "PC$"   "PC$:PRINT@773,PC$"          "PC$"   "PC$PC$PC$PC$PC$"
  "PC$"   "PC$"   "PC$:PRINT@837,PC$"          "PC$"   "PC$"
    "PC$"   "PC$"      "PC$PC$
26 PRINT@901,PC$"          "PC$"   "PC$"          "PC$"   "PC$"
     "PC$
30 PL$(1,2)=CHR$(176)+CHR$(191)+CHR$(176):PL$(2,2)=CHR$(131)+CHR
$(191)+CHR$(131):PL$(3,2)=CHR$(140)+CHR$(140)+CHR$(191):PL$(4,2)
=CHR$(191)+CHR$(140)+CHR$(140)
35 F1$=STRING$(7,188)+CHR$(26)+STRING$(7,24)+STRING$(7,191)+CHR$
(26)+STRING$(7,24)+STRING$(7,143):F2$=STRING$(4,188)+CHR$(26)+ST
RING$(4,24)+STRING$(4,191)+CHR$(26)+STRING$(4,24)+STRING$(4,143)
40 FORT=448TO505STEP3:PRINT@T,PL$(4,2);:FORX=1TO50:NEXT:PRINT@T,
PL$(4,1);:FORI=0TO200:NEXT:PRINT@T,"   ";:NEXT
50 FORT=1TO500:NEXT
```

Display instructions.

```
100 CLS:PRINTCHR$(23)STRING$(31,140):PRINTTAB(9)"T R S - M A N";
PRINTTAB(8)"BY: GEORGE DELP":PRINTSTRING$(31,140)
110 PRINTTAB(8)".  ..... 10":PRINTTAB(8)"*  ..... 100":PRI
NTTAB(7)PC$"  ..... ????":PRINTSTRING$(31,140)
120 PRINT" THE '"CHR$(93)"' KEY MAKES YOU GO LEFT":PRINT" THE '"
CHR$(94)"' KEY MAKES YOU GO RIGHT":PRINT" THE 'I' KEY MAKES YOU
GO UP":PRINT" THE '"CHR$(92)"' KEY MAKES YOU GO DOWN":PRINTSTRIN
G$(31,140)
130 PRINTTAB(5)"PRESS THE <ENTER> KEY":PRINTTAB(7)"TO START THE
GAME":PRINTSTRING$(31,140);
150 IFINKEY$<>CHR$(13)THEN150
```

Initialize variables for the game about to start.

```
500 SC=0:NF=144:GOSUB9000:NO=0
510 IFNF=144PRINT@PO,PL$(DI,1);:FORT=1TO4:PRINT@PO(T),"   ";:NEX
T:GOSUB9100:NF=0:PO=544:DI=3:PO(1)=130:PO(2)=955:PO(3)=187:PO(4)
=898:DI(1)=4:DI(2)=3:DI(3)=3:DI(4)=4:PK(1)=46:PK(2)=46:PK(3)=46:
PK(4)=46:NO=NO+1:IFNO>4NO=4
```

Display player and add to his score if necessary.

```
520 PK=PEEK(D+PO+1):IFPK=46PK=32:SC=SC+10:NF=NF+1:PRINT@0,SC;:PR
INT@PO,PL$(DI,2);
530 IFPK=42PK=32:SC=SC+100:NF=NF+1:PRINT@0,SC;:PRINT@PO,PL$(DI,2
);:GOSUB9500:CO=20:FL=1
531 CO=CO-1:IFCO=0GOSUB9510:FL=0
532 IFPO=511PO=571ELSEIFPO=574PO=514
536 C1=C1-1:IFC1=0PRINT@544,"   ";:F1=0
540 PRINT@PO,PL$(DI,1);
```

Go move the creatures chasing you.

```
545 GOSUB1000
```

Wrap-around of player going from one side of the screen to the other.

```
546 IFDI=4ANDPO=571PRINT@PO,"   ";:PO=514:GOTO510ELSEIFDI=3ANDPO
=514PRINT@PO,"   ";:PO=571:GOTO510
```

See if the player wants to change direction.

```
550 Z=PEEK(14400):IFZ<>0THEN700
```

Player doesn't change directions, so move mouth in the current direction if possible.

```
560 IFDI=1IFPEEK(D+PO-63)<129PRINT@PO,"   ";:PO=PO-64:GOTO510
570 IFDI=2IFPEEK(D+PO+65)<129PRINT@PO,"   ";:PO=PO+64:GOTO510
580 IFDI=3IFPEEK(D+PO-2)<129PRINT@PO,"   ";:PO=PO-3:GOTO510
590 IFDI=4IFPEEK(D+PO+4)<129PRINT@PO,"   ";:PO=PO+3:GOTO510
```

```
600 GOTO510
```

Player wants to change directions, so turn mouth and see if it can move.

```
700 IFZ=8THENDI=1:IFPEEK(D+PO-63)<129PRINT@PO,"   ";:PO=PO-64:GO
TO510
710 IFZ=16THENDI=2:IFPEEK(D+PO+65)<129PRINT@PO,"   ";:PO=PO+64:G
OTO510
720 IFZ=32THENDI=3:IFPEEK(D+PO-2)<129PRINT@PO,"   ";:PO=PO-3:GOT
O510
730 IFZ=64THENDI=4:IFPEEK(D+PO+4)<129PRINT@PO,"   ";:PO=PO+3:GOT
O510
740 GOTO510
```

Movement logic for the creatures who are chasing you.

```
1000 FORX=1TONO
1005 AD=PO(X)-PO
1006 IFDI(X)=4ANDPO(X)=571PRINT@PO(X)," "CHR$(PK(X))" ";:PO(X)=5
14:GOTO1500ELSEIFDI(X)=3ANDPO(X)=514PRINT@PO(X)," "CHR$(PK(X))"
";:PO(X)=571:GOTO1500
1007 IFPO(X)<520ANDPO(X)>512DI(X)=4:GOTO1100
1010 IFAD>2ANDAD<40THENIFPEEK(D+PO(X)-2)<129ORPO=PO(X)-3THENPRIN
T@PO(X)," "CHR$(PK(X))" ";:PO(X)=PO(X)-3:PK(X)=PEEK(D+PO(X)+1):P
RINT@PO(X),PC$;:DI(X)=3:GOTO1500
1020 IFAD>-40ANDAD<-2THENIFPEEK(D+PO(X)+4)<129ORPO=PO(X)+3THENPR
INT@PO(X)," "CHR$(PK(X))" ";:PO(X)=PO(X)+3:PK(X)=PEEK(D+PO(X)+1)
:PRINT@PO(X),PC$;:DI(X)=4:GOTO1500
1030 IFAD>0IFAD/64=INT(AD/64)IFPEEK(D+PO(X)-63)<129ORPO=PO(X)-64
PRINT@PO(X)," "CHR$(PK(X))" ";:PO(X)=PO(X)-64:PK(X)=PEEK(D+PO(X)
+1):PRINT@PO(X),PC$;:DI(X)=1:GOTO1500
1040 IFAD/64=INT(AD/64)IFPEEK(D+PO(X)+65)<129ORPO=PO(X)+64PRINT@
PO(X)," "CHR$(PK(X))" ";:PO(X)=PO(X)+64:PK(X)=PEEK(D+PO(X)+1):FR
INT@PO(X),PC$;:DI(X)=2:GOTO1500
1100 IFDI(X)=1IFPEEK(D+PO(X)-63)<129PRINT@PO(X)," "CHR$(PK(X))"
";:PO(X)=PO(X)-64:PK(X)=PEEK(D+PO(X)+1):GOTO1500
1110 IFDI(X)=2IFPEEK(D+PO(X)+65)<129PRINT@PO(X)," "CHR$(PK(X))"
";:PO(X)=PO(X)+64:PK(X)=PEEK(D+PO(X)+1):GOTO1500
1120 IFDI(X)=3IFPEEK(D+PO(X)-2)<129PRINT@PO(X)," "CHR$(PK(X))" "
;:PO(X)=PO(X)-3:PK(X)=PEEK(D+PO(X)+1):GOTO1500
1130 IFDI(X)=4IFPEEK(D+PO(X)+4)<129PRINT@PO(X)," "CHR$(PK(X))" "
;:PO(X)=PO(X)+3:PK(X)=PEEK(D+PO(X)+1):GOTO1500
1140 DI(X)=RND(4):GOTO1500
```

Subroutine to see if you have eaten or if you have been eaten by one of the creatures.

```
1500 IFFL=1ANDPO=PO(X)GOSUB2000:GOTO510ELSEIFPO(X)=POTHEN1501ELS
E1510
1501 PRINT@PO,PL$(1,1);:GOSUB1503:PRINT@PO,CHR$(186)CHR$(191)CHR
$(181);:GOSUB1503:PRINT@PO,CHR$(176)CHR$(191)CHR$(176);:GOSUB150
3:PRINT@PO,CHR$(176)CHR$(188)CHR$(176);:GOSUB1503:PRINT@PO,STRIN
G$(3,176);:GOSUB1503:PRINT@PO," "CHR$(176)" ";:GOSUB1503
1502 PRINT@PO,"   ";:GOSUB1503:TR=TR+1:IFTR=3PRINT@534,"YOUR SCO
RE WAS"SC;:FORT=1TO5000:NEXT:RUNELSE1515
1503 FORT=1TO200:NEXTT:RETURN
1510 PRINT@PO(X),PC$;:NEXTX:RETURN
1515 FORXX=1TONO:IFPK(XX)=170PK(XX)=32ELSEIFPK(XX)=191PK(XX)=46
1516 PRINT@PO(XX)," "CHR$(PK(XX))" ";:NEXT:PRINT@PO,"   ";
1520 PO=544:DI=3:PO(1)=130:PO(2)=955:PO(3)=187:PO(4)=898:DI(1)=4
:DI(2)=3:DI(3)=3:DI(4)=4:PK(1)=PEEK(D+PO(1)+1):PK(2)=PEEK(D+PO(2
)+1):PK(3)=PEEK(D+PO(3)+1):PK(4)=PEEK(D+PO(4)+1):GOTO510
```

Subroutine to increase your score if you have eaten one of the creatures.

```
2000 PT=RND(4)*200:PRINT@PO-1,PT;:SC=SC+PT:PRINT@0,SC;:FORI=1TO1
0:GOSUB9510:GOSUB9500:NEXT
2010 A=RND(4)
2020 IFA=1IFPEEK(D+131)<129PO(X)=130:DI(X)=4:PK(X)=PEEK(D+131):R
ETURN
2030 IFA=2IFPEEK(D+956)<129PO(X)=955:DI(X)=3:PK(X)=PEEK(D+956):R
ETURN
2040 IFA=3IFPEEK(D+188)<129PO(X)=187:DI(X)=3:PK(X)=PEEK(D+188):R
ETURN
2050 IFA=4IFPEEK(D+899)<129PO(X)=898:DI(X)=3:PK(X)=PEEK(D+899):R
ETURN
2060 GOTO2010
```

Subroutine to print the playing field on the screen.

```
9000 D1$=CHR$(188)+CHR$(26)+CHR$(24):FORT=1TO5:D1$=D1$+CHR$(191)
+CHR$(26)+CHR$(24):NEXT:D1$=D1$+CHR$(143):D2$=CHR$(188)+CHR$(24)
:FORT=1TO5:D2$=D2$+CHR$(191)+CHR$(24):NEXT:D2$=D2$+CHR$(143)
9010 D3$=CHR$(188)+CHR$(26)+CHR$(24):FORT=1TO3:D3$=D3$+CHR$(191)
+CHR$(26)+CHR$(24):NEXT:D3$=D3$+CHR$(143):D4$=STRING$(62,140):D5
$=STRING$(17,140):D6$=STRING$(14,140)
9020 D7$=CHR$(188)+STRING$(5,140)+CHR$(188)+CHR$(26)+STRING$(7,2
4)+CHR$(191)+CHR$(197)+CHR$(191)+CHR$(26)+STRING$(7,24)+CHR$(143
)+STRING$(5,140)+CHR$(143)
9030 D8$=CHR$(188)+STRING$(2,140)+CHR$(188)+CHR$(26)+STRING$(4,2
4)+CHR$(191)+"  "+CHR$(191)+CHR$(26)+STRING$(4,24)+CHR$(143)+STR
ING$(2,140)+CHR$(143)
9040 CLS:PRINT@64,D1$;:PRINT@576,D1$;:PRINT@127,D2$;:PRINT@639,L
EFT$(D2$,11);:POKE16383,143:PRINT@210,D3$;:PRINT@216,D3$;:PRINT@
231,D3$;:PRINT@237,D3$;:PRINT@594,D3$;:PRINT@600,D3$;:PRINT@615,
D3$;:PRINT@621,D3$;
9050 PRINT@65,D4$;:PRINT@961,D4$;:PRINT@449,D5$;:PRINT@577,D5$;:
PRINT@494,D5$;:PRINT@622,D5$;:PRINT@473,D6$;:PRINT@601,D6$;:PRIN
T@198,D7$;:PRINT@710,D7$;:PRINT@243,D7$;:PRINT@755,D7$;:PRINT@22
2,D8$;:PRINT@734,D8$;
9060 RETURN
```

Subroutine to print all the dots and asterisks on the screen.

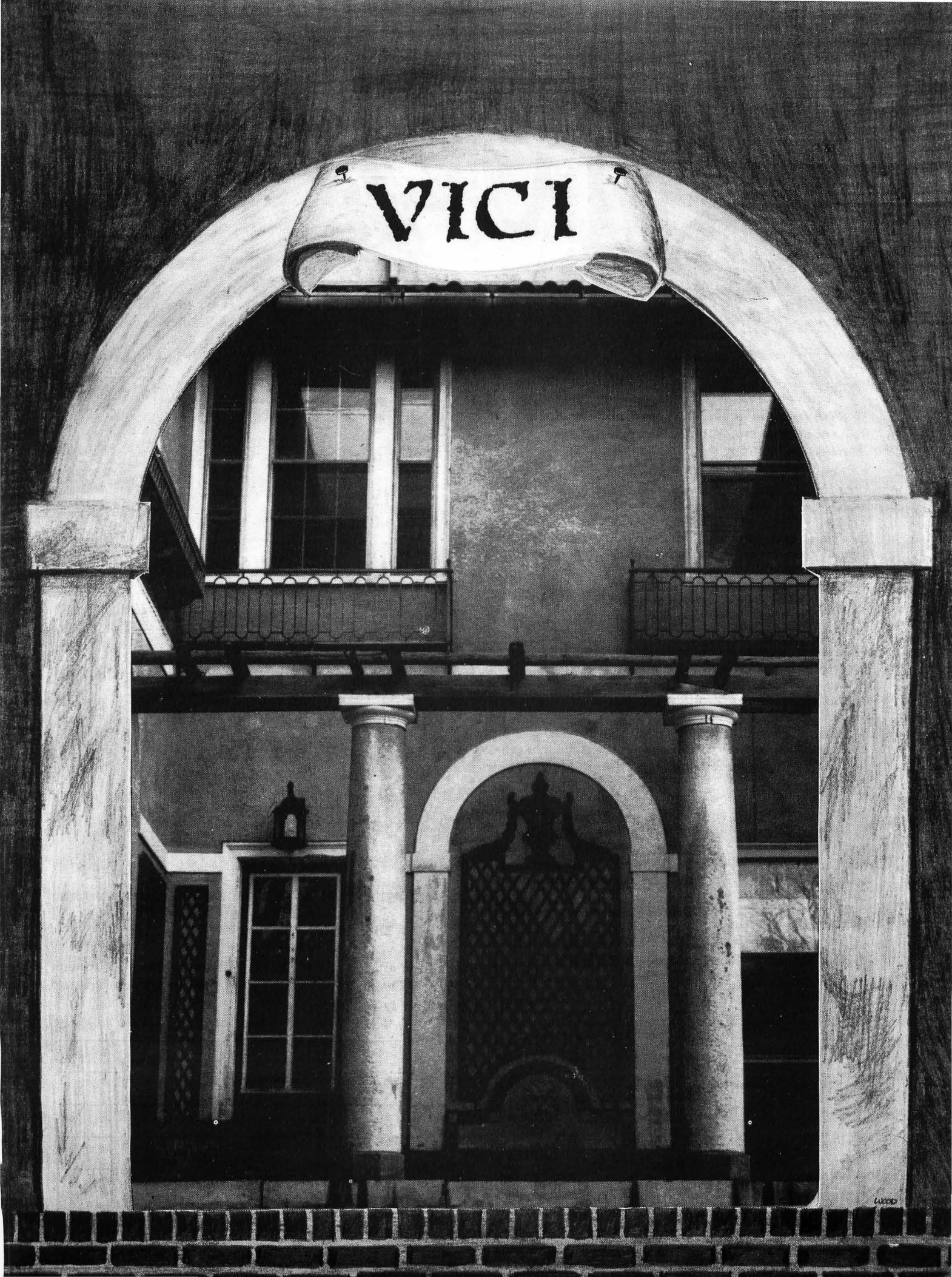
```
9100 RESTORE:PRINT@22,"T  R  S  -  M  A  N";
9110 READA,B:IFA=-1THEN9120ELSEPRINT@A,"";:FORT=1TOB:PRINT".  ";
:NEXT:GOTO9110
9120 READA,B:IFA=-1THEN9130ELSEFORT=ATOBSTEP64:PRINT@T-1," . ";:
NEXT:GOTO9120
9130 PRINT@393,"*";:PRINT@438,"*";:PRINT@649,"*";:PRINT@694,"*";
9140 RETURN
9200 DATA131,20,515,20,899,20,387,5,643,5,411,4,667,4,432,5,688,
5,-1,-1,195,323,207,335,219,411,228,420,240,432,252,444,707,835,
719,847,731,859,740,868,752,880,764,892,213,853,234,874,-1,-1
```

Subroutine to light up all the squares on the screen.

```
9500 PRINT@198,F1$;:PRINT@710,F1$;:PRINT@243,F1$;:PRINT@755,F1$;
:PRINT@222,F2$;:PRINT@734,F2$;:RETURN
```

Subroutine to turn off all the squares on the screen.

```
9510 PRINT@198,D7$;:PRINT@710,D7$;:PRINT@243,D7$;:PRINT@755,D7$;
:PRINT@222,D8$;:PRINT@734,D8$;:RETURN
```


VICI

The Captain 80 Book of

BASIC *Adventures*

- Eighteen famous adventures, listed in large easy to read type, ready to be keyed in. Adventures by Boner, Kepner, Powers, Micklus, Forsythe, Greg Hassett and others.
- Includes *Temple of the Sun, Dog Star, Lost Ship, Spider Mountain, Lost Dutchman's Gold, Thunder Road, Sorcerer's Castle, Deadly Dungeon, Atlantean Oddysey* and others. These adventures would cost over $200 is purchased individually!
- Includes a unique *Adventure Generator* program - not available anywhere. YES, this program will actually write another BASIC adventure program, which you may then run. Not even the author of the generator program knows the outcome! *This program alone is worth the price of the entire book!*
- Includes chapters on what an adventure is, how to play adventures, how to write adventures, how to sell your adventure, ten adventure program ideas, and more!
- Study the techniques and methods used by the masters of adventure program writing. All adventure programs listed in this book are in their original form and in full length. Although specifically written for the TRS-80 Model I and III, these programs are adaptable to other computers using Microsoft's BASIC.
- Available at your dealer approximately November 15th, 1981. The suggested retail price of this book is $19.95.
- The BASIC Adventure Book was compiled and edited by Bob Liddil, of The Programmers Guild. It measures 8½ by 11 inches and contains 256 pages.

A neat Christmas gift idea!

- *ORDER YOURS TODAY!*

Dealer discounts available - please inquire

TRS-80® is a registered trademark of the Tandy Corp.

Send To: 80-U.S. Journal
3838 South Warner Street
Tacoma. Washington 98409

Please send my copy of the BASIC Adventure Book. My check or Money Order for $22.00 ($19.95 plus $2.05 postage and handling) is enclosed.

Name_____

Address _____

City_____ State_____ Zip _____

Visa/MC _____

Exp. Date_____ Signature _____

(206) 475-2219 Phone orders accepted for Visa/MC only.

Order#SS

VICI

By ROBERT J. POLLOCK

***Vici* is a TRS-80® political war game for one player requiring 16K Level II BASIC, 16K Model III BASIC, or 32K Disk BASIC.**

The Cessa Valley is surrounded by mountains formidable enough to make it nearly impenetrable. The two small states the valley contains, Vici and Arce, are threatened only by each other.

Your older brother was the ruler of Vici. For his peace of mind and your safety, you were sent into exile when very young. Now, he has died childless. So you as heir presumptive, will be offered the throne.

All you know of the state affairs of Vici is that the major decisions are made at the semi-annual courts. In the fall at the Harvest Court, the ruler sells as much as he wants of his reserves to the assembled merchants and plans for next year's farms. In the spring at the Military Court, the ruler plans for the defense of the realm and any other military objectives as he desires and can afford. Armies are very expensive!

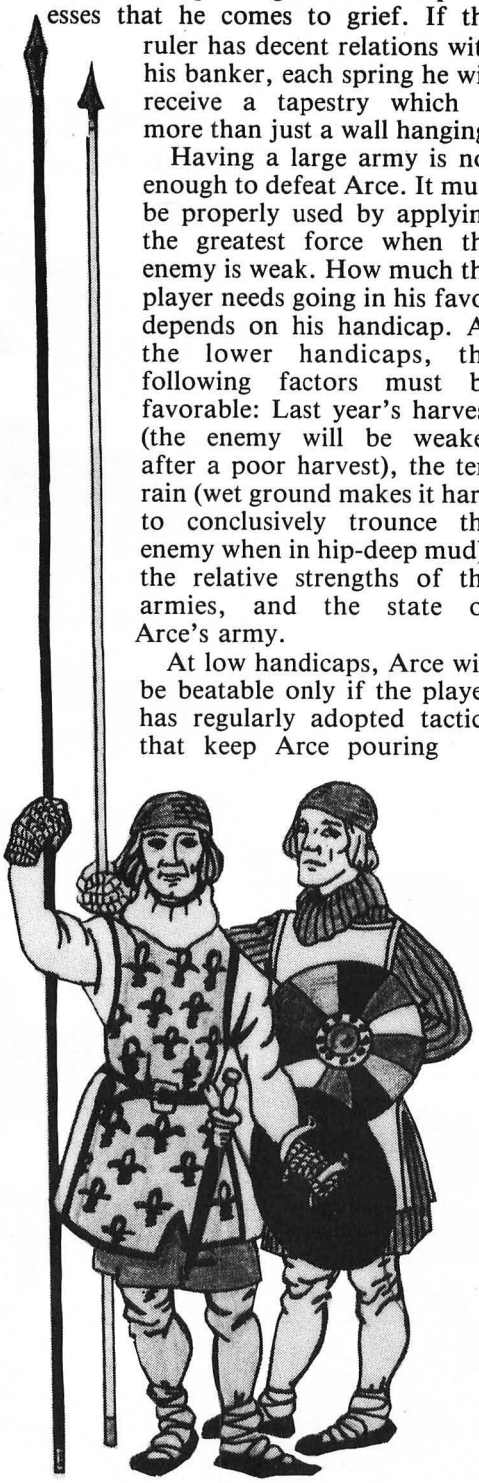
Peace is dear to all wise rulers, however Vici and Arce have been warring as long as anyone can remember. The only way to end the strife is for one to conquer the other because the game does not permit peace. The player is not forced to conquer Arce, but Vici will suffer depredations until Arce falls.

The way to defeat Arce is to conquer its army with yours. That requires cash. The primary source of cash is Vici's agricultural output. The output is determined by the weather and the attitudes of the population. The weather does not, in itself, help or hurt either side. However, the changes from year to year permit the ruler to profit from or lose in speculations. Concerning attitudes, the game gives the player quite a bit of latitude. No single act, no matter how unjust, dooms the ruler because internal events are the outcome of processes, not acts.

Crime can serve as an example. The program postulates an insufficient number of criminals to do much damage. If, however, the ruler makes decisions that force people off farms, some of them will turn to crime. Criminals can be caught and a large enough army will discourage some instances of criminality, but that costs, and it is not enough to save a ruler firmly committed to foolish policies. The first task of the player then is to extract as much cash as possible without so upsetting the internal processes that he comes to grief. If the ruler has decent relations with his banker, each spring he will receive a tapestry which is more than just a wall hanging.

Having a large army is not enough to defeat Arce. It must be properly used by applying the greatest force when the enemy is weak. How much the player needs going in his favor depends on his handicap. At the lower handicaps, the following factors must be favorable: Last year's harvest (the enemy will be weaker after a poor harvest), the terrain (wet ground makes it hard to conclusively trounce the enemy when in hip-deep mud), the relative strengths of the armies, and the state of Arce's army.

At low handicaps, Arce will be beatable only if the player has regularly adopted tactics that keep Arce pouring her resources into replacing soldiers rather than into reserves. In contrast, at the highest handicap, Arce is a pushover. If its first harvest is poor, invading with a good-sized army will bring victory before the first year of the reign is over.



The ruler, then, can fail if he lacks the resources and knowledge to act at the right moment. The same rules, in general, apply to the conquest of Vici, except Arce is always assigned the lower handicap.

A ruler may fail because he runs out of time. The higher the handicap, the greater the period of time he is allowed, but eventually, an unconquering ruler is gently retired.

Instructions

Playing *Vici* does not require any special skills or knowledge. The world it models is not particularly esoteric (nor plausible) and devoid of supernatural elements.

Much of the program is devoted to screen displays and keyboard inputs. If the program needs a response from you, it will let you know. Most of the time you respond to the video display by hitting a single key. "Please indicate . . ." means the program is waiting for your order to continue. Any regular key will continue the program. "(Y/N)?" means the program needs a yes or no answer. Just hit the "Y" or "N" key as appropriate. The single stroke inputs mean you have to watch your fingers. Resting them heavily on the keyboard can depress a key. The program assumes you can foretell the next response needed. It stores a keypress and acts on it if the next needed response is of the single stroke input type.

From time to time, the program will request a number from you. These are the only times when it is necessary to use the ENTER key. Type in the number you want and hit ENTER. The game begins with such a situation by asking for your handicap. The larger the number you select, the easier the game.

Occasionally, the program gives you messages that do not require a

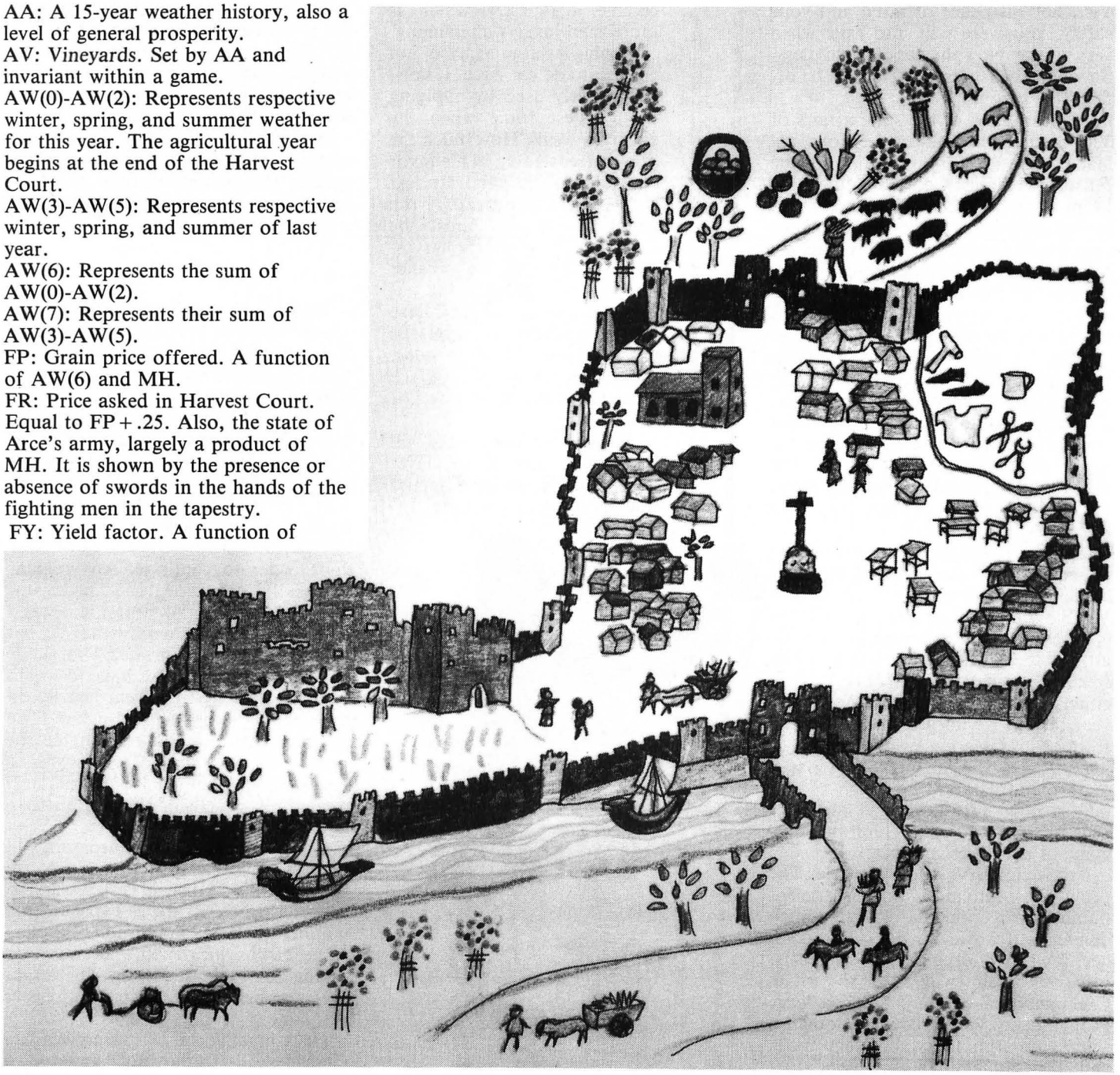
response. The program will move on its own after a time. You can change most of these pauses by changing program line 3800. It now begins: FORXO =0TO200... If you change the 200 to a larger number, the pauses will lengthen and a smaller number will shorten them.

A note on currency: 100 coppers = one silver piece.

There is no save-game feature. While *Vici* has puzzle elements, it is not a puzzle. Because significant events are the outcome of processes, there is no point, short of victory, where the player is safe. Good luck.

Variables

AA: A 15-year weather history, also a level of general prosperity.
AV: Vineyards. Set by AA and invariant within a game.
AW(0)-AW(2): Represents respective winter, spring, and summer weather for this year. The agricultural year begins at the end of the Harvest Court.
AW(3)-AW(5): Represents respective winter, spring, and summer of last year.
AW(6): Represents the sum of AW(0)-AW(2).
AW(7): Represents their sum of AW(3)-AW(5).
FP: Grain price offered. A function of AW(6) and MH.
FR: Price asked in Harvest Court. Equal to FP + .25. Also, the state of Arce's army, largely a product of MH. It is shown by the presence or absence of swords in the hands of the fighting men in the tapestry.
FY: Yield factor. A function of AW(6). It determines the productivity of the land and the sheep herd.
ME: Arce's reserves. At the highest handicap, it has little significance, but as the play becomes more difficult, it may mandate a change in strategy.
ME(1): Arce's army.
MH: The player's handicap.
MM: Financial standing. Determines whether the banker will send a tapestry.
MP: A disposition to resentment (rents).
MR: A disposition to distrust.
MT: Equal to MP + MV.
MV: A disposition to violence.
MX: Farmers killed.
MY: Time. The years of the reign. It is diplayed on the victory graphics.
MZ: Farmers plundered.
PF: Farms. A product of many elements and a factor in others.
PG: Vici's army.
PV: Criminal vagrants. A powerful factor in crime.
S(2): Grain.
S(3): Sheep.
S(4): Wine.
TA: Credit available.
TB: Debt.
TR: Cash.
T(1): Rent. The percent of the harvest owed to the ruler.
T(6): Interest. The rate at which the ruler can borrow money. It varies with AA and debt status.



```
$$$$$$$$$$$$$$$$$$$$$$$$
$      S-80  BASIC      $
$        'VICI'         $
$AUTHOR:  Robert Pollock$
$   (c) 1982 SoftSide   $
$$$$$$$$$$$$$$$$$$$$$$$$
```

Meta-game. Clear string space, define variables, obtain handicap, etc.

```
150 RANDOM:CLEAR250:DEFSTRH,I,Y:DEFINTA,M,P,X:DIMAA(14):CLS:INPU
 T"YOUR HANDICAP (1 TO 10)";Q:MH=Q:IFMH>10ORMH<1ORMH<>QGOTO150ELS
 ECLS:XL=0:A=86:GOSUB4100:GOSUB4500:FORX=0TO2:GOSUB4200:NEXT:GOSU
 B4100
160 IFMH<3FP(0)=0ELSEIFMH>2ANDMH<5THENFP(0)=.25ELSEIFMH>4ANDMH<7
 FP(0)=.5ELSEIFMH>6ANDMH<9FP(0)=.75ELSEFP(0)=1
```

Declaration of string variables.

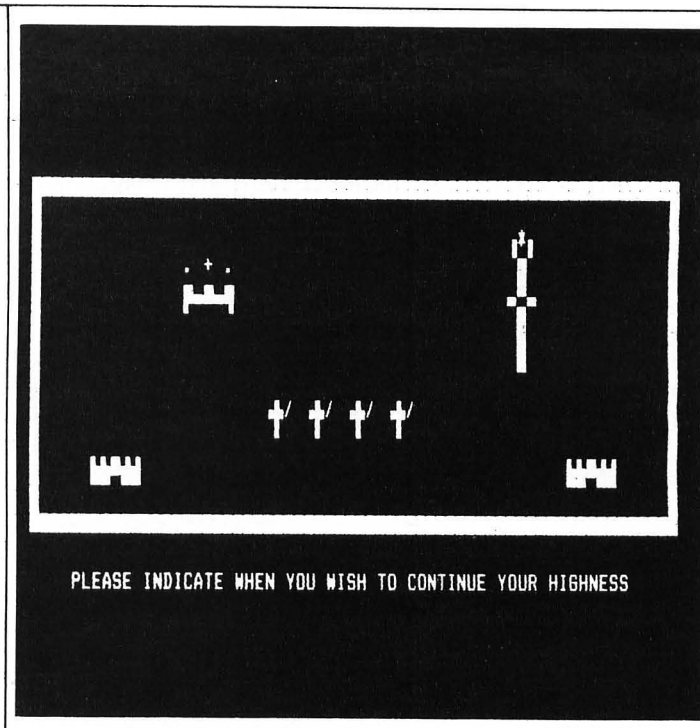
```
 200 T$="TOTAL":YH="YOUR HIGHNESS":TT$="    ":H(2)="GRAIN":H1(2)=
 "BUSHELS":H(3)="SHEEP":H1(3)="HEAD":H(4)="WINE":H1(4)="CASKS":RP
 $=" REPORT":HD(0)="CROWN ":HD(1)="RESERVES"
210 HD(2)="RESOURCES":HD(3)="HARVEST":HD(5)="SPRING":HD(4)="WINT
 ER":HD(6)=" COURT":Y(0)="PRODIGAL!":Y(1)="AMPLE.":Y(2)="MEAGER."
 :Y(3)="BAD!":TH$="THE":H1="FARMS":CD$="COMMANDANT'S"+RP$+": "
220 H2(0)="RENTED":H2(1)="FALLOW":PG$="MEN-AT-ARMS":SL$="SILVER
 PIECES":H2(2)="VINEYARDS":PL$="PLUNDERED":YN=" (Y/N)?":DW$=" WIL
 L YOU "
230 DD$="DO YOU WISH TO ":DA$(0)="SELL ":DA$(1)="BUY ":CP$="COPP
 ERS/":MM$(2)="BUSHEL":MM$(3)=H1(3):MM$(4)="CASK":PU$="######":IS
 ="".  IS THAT SATISFACTORY"+YN
```

Set up initial conditions and open game.

```
300 T(1)=.25:S(1)=0:TR=0:PV=RND(35):IFPV<25GOTO300
 310 PRINT@704,"THE RULE OF VICI IS YOURS OF RIGHT.  DO YOU TAKE
 INTO YOUR CARE  THE HONOR OF YOUR FAMILY AND THE FORTUNES OF THI
 S LAND"+YN:GOSUB3400:IFIN="N"CLS:END
320 PRINT@899,"GODSPEED YOUR JOURNEY TO OUR BELOVED ANCESTRAL HO
 MELAND!":FORXX=0TO5:GOSUB1000:NEXT
330 GOSUB3700:PRINT@512,TT$+"WE, YOUR LOYAL SUBJECTS, REJOICE IN
  YOUR ARRIVAL.  YOUR":PRINT"REIGN AND DUTIES BEGIN.  WE KNOW GOO
 D FORTUNE ATTENDS US ALL."
340 GOSUB3800:FORXX=6TO13:GOSUB1000:NEXT:IFAA<150RAA>396GOTO340
 350 TR=750-(AA*10):T(6)=(10+AW(6))/100:TB=-AA*50:AV=AA:PF=3000+(
 60*(23-AA)):PJ=100000/AA:S(3)=PJ:PG=PF/100*10:PJ=PF*1.4:S(2)=PJ:
 ME=400-(8*MH):GOSUB1700
```

Main program body. It calls subroutines and performs some logical and arithmetic functions. If the game is not ended, line 500 sends the program to line 400.

```
400 GOSUB1900:GOSUB1800:GOSUB1600:GOSUB2000:IFTB<0GOSUB2100:GOSU
 B2400:ELSEGOSUB2400:GOSUB2100
410 GOSUB4000:GOSUB1000:GOSUB2600:IFPF<185060TO3200
 420 MB=0:MX=0:ME=ME+FY*(30-3*(MH-1)):A$="*":GOSUB3100:PRINT@529,
 "THE "+HD(4)+" SNOWS WERE "+Y(AW(3)+1):XL=PG*1.13-ME(1):ME(1)=PG
 *1.13:IFXL<METHENME=ME-XLELSEME=0:ME(1)=PF*.12
430 FR=RND(10):IFFR<3FR=0ELSEIFFR>MH+4FR=0ELSEFR=1
 440 PG(1)=0:ME(2)=0:GOSUB3500:IFMM<2GOSUB3700:PRINT@512,TH$+" BA
 NKER HAS SENT A TAPESTRY.  "+DD$+"SEE IT"+YN:GOSUB3400:IFIN="Y"G
 OSUB4300
450 CLS:HD="MILITARY"+HD(6):GOSUB3900:MS=12:CLS:XL=0:A=91:GOSUB4
 100:GOSUB4500:GOSUB4200:GOSUB4200:GOSUB4100
```




```
460 PRINT"TACTICAL ALTERNATIVES;":PRINT,"1)  RESPOND AND DEFEND,
 ":PRINT,"2)  RAID AND HARASS,":PRINT,"3)  INVADE ARCE."
470 XL=14:PRINT@XL*64,YH+", WHICH ALTERNATIVE WILL YOU ADOPT";:I
 NPUTQ:AO=Q:IFAO<1ORAO>3ORAO<>QGOSUB2280:GOTO470
480 ME(0)=(PF*(1+AO/(10-AO)))/10:GOSUB1500:MS=10:MB=1:GOSUB4000:
 A$=".":GOSUB3100:PRINT@529,"THE "+HD(5)+" RAINS WERE "+Y(AW(4)+1
 ):GOSUB3500:IFPG<PF/18GOTO3200ELSEIFAO=3GOSUB2700ELSEIFAO=2GOSUB
 2800ELSEGOSUB2900
490 ME(0)=PF*.11:PG=PG-PG(1):ME(1)=ME(1)-ME(2):IFMC<500RPG<PF/18
 GOTO3200ELSEIFPG(1)>0GOSUB1500
500 IFMY>54-MHGOSUB3700:PRINT@518,"OUR RESPECT IS UNDIMMED, BUT
 YOUR ADVANCED AGE WILL SOON  END YOUR ABILITY TO RULE.  YOU ARE
 DEPOSED!":FORX=1TO15:GOSUB3800:NEXT:CLS:END:ELSEGOTO400
```

Subroutines. All subroutines begin on a line number that is a multiple of 100.

Weather.

```
1000 FORX=0TO2:AW(X)=AW(X+3):NEXT:FORX=3TO5:AW(X)=RND(2)-1:NEXT:
 AW(7)=AW(0)+AW(1)+AW(2):AW(6)=AW(3)+AW(4)+AW(5):AA(14)=AW(6):AA=
 0:FORX=0TO13:AA(X)=AA(X+1):AA=AA+AA(X):NEXT:AA=AA+AW(6)
 1010 IFAW(6)=0FY=1.4:FP=4.5:ELSEIFAW(6)=1FY=1.2:FP=4.75:ELSEIFAW
 (6)=2FY=1:FP=5.25:ELSEFY=.8:FP=6
 1020 FP=FP+FP(0):RETURN
```

Crime.

```
1100 MV=MV-1+AA/6-(PG-(PF*.09))/10:IFMV<1MV=1
 1110 MZ=20*(PV*300/PF+MV+AW(7)):MX=(MZ*(MV+AW(7))/30)+MA:MW=(PV+
 MV+AW(7))*10:PF=PF-MX:S(3)=S(3)-MW
 1120 MX(0)=MX/(AW(7)+2):MZ(0)=MZ/(AW(7)+4):MW(0)=MW/(25+MV+AW(7)
 ):PV=PV-((MX(0)+MZ(0)+MW(0))/4):IFPV<PF/120PV=PF/120
 1130 MT=MV+MP:RETURN
1200 PRINT@XL*64,H(2),,:PRINTUSINGPU$;S(2);:PRINT" "+H1(2):XL=XL
 +1:RETURN
1300 PRINT@XL*64,"EXCHEQUER",,:PRINTUSINGPU$;TR;:PRINT" "+SL$:XL
 =XL+1:IFTB<0PRINT@XL*64,"DEBT",,:PRINTUSINGPU$;TB;:PRINT" "+SL$+
 " @";T(6)*100;"%":XL=XL+1
```

```
1310 IFTA>0PRINT@XL*64,"CREDIT AVAILABLE",;PRINTUSINGPU$;TA;:PRI
NT" "+SL$+" @";T(6)*100;"%":XL=XL+1
1320 RETURN
1400 PRINT@XL*64,H(3),,:PRINTUSINGPU$;S(3);:PRINT" "+H1(3):XL=XL
+1:RETURN
```

Army recruitment.

```
1500 CLS:GOSUB3600:XL=2:A=42:GOSUB4100:PRINT@192,CD$+" ARMY RECR
UITMENT.":XL=4:A=61:GOSUB4100:IFMS=10PRINT@337,"PRESENT";ELSEPRI
NT@336,"LAST YEAR";
1510 PRINTTAB(31)"RECOMMENDED";TAB(52)"COST/MAN":PRINTPG$,:PRINT
USINGPU$;PG;:PRINT,:PRINTUSINGPU$;ME(0);:PRINTTAB(47)MS;SL$:XL=7
:A=45:GOSUB4100:XL=XL+1:GOSUB1300:A=42:GOSUB4100:XL=XL+1
1520 PRINT@XL*64,"HIGHNESS, HOW MANY ";:IFMS<>12PRINT"ADDITIONAL
";
1530 PRINTPG$+DW$+"HAVE";:INPUTQ:AB=Q:IFQ*MS>TA+TRTHENXN=1:GOSUB
2290:GOTO1520:ELSEIFQ<0ORQ<>ABGOSUB2280:GOTO1520:ELSEIFQ=0GOTO15
50
1540 PRINT@(XL+1)*64,"COST:";Q*MS;SL$+IS:GOSUB3400:IFIN="N"GOSUB
2280:GOTO1520
1550 TR=TR-Q*MS:IFTR<0THENTA=TA+TR:TB=TB+TR:TR=0
1560 IFMS=12PG=Q:RETURN:ELSEPG=PG+Q:RETURN
```

Land use and rents.

```
1600 CLS:GOSUB3600:XL=2:A=42:GOSUB4100:PRINT@192,"LAND USE:":XL=
4:A=61:GOSUB4100:PRINT@320,H1,,:PRINT" "+H2(0)," "+H2(1):PRINTTT
$+"THIS YEAR",,:PRINTUSINGPU$;PF(0);:PRINT,:PRINTUSINGPU$;4000-P
F(0):PRINTTT$+"NEXT YEAR",,
1610 PRINTUSINGPU$;PF;:PRINT,:PRINTUSINGPU$;4000-PF:PRINT,"SEED
REQUIRED:";PF*4;H1(2):A=45:XL=9:GOSUB4100:PRINT@640,H2(2),,AV:XL
=11:A=42:GOSUB4100:S(2)=S(2)-PF*4
1620 XL=12:GOSUB2280:PRINT@XL*64,"ACCEPTED RENT: 25% OF "+TH$+"
"+HD(3)+IS:GOSUB3400:T(1)=.25:IFIN="N"PRINT@(XL+1)*64,"WHAT % "+
DD$+"CHARGE";:INPUTT(1):AB=T(1):IFAB<>T(1)GOTO1620ELSET(1)=T(1)/
100
1630 IFT(1)<0ORT(1)>1GOSUB2280:GOTO1620
1640 IFT(1)>.25+3*(3-AW(6))/100MP=MP+2:MV=MV+1+(T(1)-.25)*20
1650 RETURN
1700 CLS:PU=31:HD=HD(0)+HD(1):GOSUB3600:XL=2:A=42:GOSUB4100:XL=X
L+2:GOSUB1200:A=45:GOSUB4100:XL=XL+1:GOSUB1400:GOSUB4100:XL=XL+1
:GOSUB4600:XL=XL+1:A=42:GOSUB4100:GOSUB3500:RETURN
```

Harvest and population change.

```
1800 TA=0:CLS:GOSUB3600:MY=MY+1:MA=100*FY:MB=T(1)*MA:MC=MA-MB:IF
MC<60MC=MC+(60-MC)/2:MB=MA-MC:PD=-(PF*((60-MC)/60)):PV=PV-PD/20:
MR=MR+2:MP=MP+2:GOTO1820
1810 IFMR<2ANDMP<2IFPF=4000PD=0ELSEPD=(4000-PF)/10ELSEIFMP+MR>3P
D=PF*.04ELSEPD=PF*.03
1820 MA=0:FR=FP+.25:PJ=S(3)*.5*FY:D(3)=PJ:D(4)=AV*FY*150:D(2)=MB
*(PF-MZ-MZ(1)):PF(0)=PF:PF=PF+PD:IFPF>4000PV=(PF-4000)/5+PV:PF=4
000
1830 IFD(3)+S(3)>12500THEND(3)=12500-S(3)ELSEIFD(3)<1THEND(3)=0:
S(3)=0
1840 XL=2:A=42:GOSUB4100:PRINT@192,TH$+" "+HD(3)+" WAS "+Y(AW(6)
):XL=4:A=61:GOSUB4100:FORX=2TO4:PRINT@(X+3)*64,H(X),,:PRINTUSING
PU$;D(X);:IFX=3PRINT" LAMBS"ELSEPRINT" "+H1(X)
1850 NEXT:XL=8:A=45:GOSUB4100:PRINT@576,"NEW "+T$+"S":FORX=2TO4:
S(X)=S(X)+D(X):PRINT@(X+8)*64,TT$+H(X),,:PRINTUSINGPU$;S(X);:PRI
NT" "+H1(X):NEXT:S(1)=PF*4:S(0)=S(2)-S(1):XL=13:A=42:GOSUB4100:G
OSUB3500:RETURN
1900 GOSUB1100:CLS:HD=HD(3)+HD(6):GOSUB3900:CLS:PU=31:GOSUB3600:
XL=2:A=42:GOSUB4100:PRINT@192,CD$+"ANNUAL SUMMARY OF CAPITAL CRI
MES."
1910 XL=4:A=61:GOSUB4100:PRINT@320,"CRIME",,"  NUMBER","APPREHEN
DED":A=58:XL=6:GOSUB4100:PRINT@448,"FARMS"
1920 PRINT@512,TT$+PL$,,:PRINTUSINGPU$;MZ;:PRINT,:PRINTUSINGPU$;
MZ(0);:PRINT,TT$+"WITH FATALITES",:PRINTUSINGPU$;MX;:PRINT,:PRIN
TUSINGPU$;MX(0)
1930 XL=10:A=45:GOSUB4100:PRINT@704,H(3)+" STOLEN",,:PRINTUSINGP
U$;MW;:PRINT,:PRINTUSINGPU$;MW(0):XL=12:A=42:GOSUB4100:GOSUB3500
:RETURN
```

Disaster and relief.

```
2000 Q=(MZ+MZ(1))*50:IFQ>S(2)IN="N":GOTO2010:ELSEGOSUB3700:PRINT
@512,TT$+TH$+" "+H1+" THAT WERE "+PL$+" NEED "+H(2)+" TO SURVIVE
 UNTIL  "+TH$+" NEXT "+HD(3)+" (";Q;+H1(2)+")."+DW$+"SUPPLY IT"
+YN:GOSUB3400
2010 IFIN<>"Y"PF=PF-MZ-MZ(1):PV=PV+(MZ+MZ(1))/5:RETURN
2020 S(2)=S(2)-Q:MP=MP-1:MR=MR-1:IFMP<0MP=0
2030 IFMR<0MR=0
2040 RETURN
```

Merchant.

```
2100 FORXM=2TO4:CLS:PRINTYH+": "+TH$+" "+H(XM)+" MERCHANT.":PRIN
T"HERE ARE YOUR RECORDS, SHOULD YOU WISH TO CONSULT THEM.":A=42:
XL=2:GOSUB4100:XL=XL+2:ONXM-1GOSUB1200,1400,4600
2110 A=45:GOSUB4100:XL=XL+1:GOSUB1300:XL=XL+1:A=42:GOSUB4100:XL=
XL+2
2120 A=128:FORXN=0TO1:IFS(XM)>0ORXN=1GOSUB2310:IFIN="Y"GOTO2150
2130 NEXTXN:NEXT:GOSUB2280:PRINT@(XL+1)*64,"HAVE YOU FURTHER BUS
INESS WITH THE MERCHANTS HIGHNESS (Y/N)?":GOSUB3400:IFIN="Y"GOTO
```

```
2100
 2140 RETURN
2150 ONXM-1GOSUB2160,2180,2200:GOTO2220
 2160 FQ=FP:IFXN=1FQ=FR
2170 RETURN
2180 MQ=1/FY*15:FQ=MQ:IFXN=1MQ=MQ+(MQ*.25):FQ=MQ
 2190 RETURN
2200 MQ=1/FY*10:FQ=MQ:IFXN=1MQ=MQ+(MQ*.15):FQ=MQ
 2210 RETURN
2220 PRINT@XL*64,YH+", THE PRICE IS";FQ;CP$+MM$(XM)+".":XL=XL+1
2230 PRINT@XL*64,"HOW MANY "+H1(XM)+" MIGHT "+YH+" "+DA$(XN);:IN
PUTQ:IFQ>0ANDQ<256GOTO2300ELSEIFQ<0GOSUB4100:GOTO2230:ELSEIFQ=0XL
=XL-1:GOSUB2280:GOTO2130:ELSEIFQ>S(XM)ANDXN=0GOSUB2290:GOTO2230
 2240 XX=FQ*Q/100:IFXX>32500.9PRINT@(XL+1)*64,YH+", WE CAN'T HAND
LE THAT LARGE A TRANSACTION":GOSUB3800:GOSUB2280:GOTO2230
 2250 PP=XX:IFXN=1PP=-PP:Q=-Q:IFPP+TR<0XL=XL-2:GOSUB2290:GOTO2230
2260 PRINT@(XL+1)*64,TH$+" "+T$+" IS";ABS(PP);SL$+IS:GOSUB3400 :
IFIN="N"GOSUB2280:GOTO2230
2270 XL=XL-1:XN=1:TR=TR+PP:S(XM)=S(XM)-Q:IFS(XM)<1S(XM)=0:GOTO21
30:ELSEGOTO2130
2280 PRINT@XL*64,CHR$(31):RETURN
 2290 PRINT@(XL+2)*64,"I BELIEVE THAT EXCEEDS "+YH+"'S "+HD(XN+1)
:GOSUB3800:GOSUB2280:RETURN
2300 PRINT@(XL+2)*64,YH+", WE CAN'T DEAL IN SO SMALL A QUANTITY.
":FORXO=0TO600:NEXTXO:GOSUB2280:GOTO2230
2310 PRINT@XL*64,DW$+DA$(XN)+H(XM)+" "+YH+YN:GOSUB3400:GOSUB2280
:RETURN
```

Banker.

```
2400 CLS:GOSUB3600:XL=2:A=42:GOSUB4100:XL=3:PRINT@XL*64,"TREASUR
Y"+RP$+":":XL=4:A=61:GOSUB4100:XL=5:XN=-T(6)*TB:TB=TB-XN:GOSUB13
00:A=42:GOSUB4100:XL=XL+2:PP=XL:IFTB=0GOTO2480
 2410 IFXN<26GOTO2430
2420 PRINT@XL*64,"INTEREST DUE:";XN;SL$+".":XL=XL+1
 2430 PRINT@XL*64,YH+", HOW MUCH"+DW$+"PAY ON THE DEBT";:INPUTQ:A
B=Q:IFQ>TRTHENXN=1:GOSUB2290:XL=PP:GOTO2410:ELSEIFQ<0ORAB<>0GOSU
B2280:GOTO2430
2440 IFQ>-TBPRINT"THAT EXCEEDS YOUR DEBT, "+YH:FORXA=1TO600:NEXT
XA:XL=PP:GOSUB2280:GOTO2410
2450 TR=TR-Q:TB=TB+Q:IFQ<XNTHENMM=MM+1ELSEMM=MM-1
 2460 IFMM<0THENMM=0
2470 IFMM>0RETURN
2480 XL=PP:GOSUB2280:TA=PF+TB:IFTA<250GOTO2510ELSEPRINT@XL*64,YH
+", THE BANKER OFFERS TO ";:IFTB<0PRINT"REFINANCE YOUR DEBT @";T
(6)*100+1;"%","AND EXTEND";TA;SL$+" OF CREDIT ON THE SAME TERMS.
"
2490 IFTB>-1XX=AA/4:T(6)=XX/100:PRINT"EXTEND";TA;SL$,"OF CREDIT
@";T(6)*100+1;"%.  ";
2500 PRINTDD$+"ACCEPT"+YN:GOSUB3400:IFIN="Y"T(6)=T(6)+.01:RETURN
 2510 TA=0:RETURN
```

Riot.

```
2600 IFMV<20RPG=>PF/100*(9+MT)THENMA=0:RETURN:ELSEA$="RIOT    ":
A$=A$+A$+A$+A$:CLS:PRINTCHR$(23):FORXL=0TO15:PRINT@XL*64,A$:NEXT
XL:CLS:GOSUB3800:GOSUB3700:MA=MT*10+PV:MP=MP-1:AP=MA/24
 2610 IFAP<2AP=2
2620 PF=PF-MA:PG=PG-AP:PRINT@515,AP;PG$+" AND";MA;"FARMERS WERE
KILLED IN THE RIOTING!":GOSUB3500:RETURN
```

Outcome of the season's military activities.

```
2700 GOSUB3700:PRINT@512,"THE ARMY IS READY TO INVADE ARCE! "+DW
$+"HAVE IT SO"+YN:GOSUB3400:IFIN="N"GOSUB2900:RETURN
 2710 CLS:GOSUB3800:GOSUB3800:IFAW(7)>1IFPG>ME(1)+MEANDAW(0)+AW(1
)=2ANDFR=1ORMH>2ANDPG>ME(1)+MEANDAW(0)+AW(1)=2ORMH>5ANDAW(0)=1AN
DPG>1.1*ME(1)ORMH>8ANDPG>1.1*ME(1)GOTO3300
2720 ME(2)=10*(AW(3)+AW(4)+1.5*FR+AW(7)/2)*PG/ME(1)-RND(15):IFME
(2)<4ME(2)=5+RND(10)
2730 PG(1)=(ME(1)+PG)/12-ME(2):GOSUB3700:PRINT@522,"WE KILLED";M
E(2);"AND LOST";PG(1);PG$:GOSUB3500:RETURN
2800 IFMH<6GOSUB3000
2810 Q=RND(4)+1:ME(2)=Q*(3*AW(3)+AW(4)+AW(7)+3*FR+2):FORX=1TOQ:P
G(1)=PG(1)+RND(10):NEXT:GOSUB3700:PRINT@512,"WE MADE";Q;"RAIDS I
NTO ARCE.  WE KILLED";ME(2);"AND LOST";PG(1);PG$:GOSUB3500:RETUR
N
2900 GOSUB3000:Q=3+RND(AW(6)+1):FORX=1TOQ:ME(2)=ME(2)+2.5*(5-AW(
6)):NEXT:ME(2)=ME(2)-MT-MR:PG(1)=(MT+MR)*2*Q+RND(10):MZ(1)=Q*(16
+AW(6)):GOSUB3700:PRINT@515,"WE SUFFERED";Q;"RAIDS.  WE KILLED";
ME(2);"AND LOST";PG(1);PG$+"."
2910 PRINT"THEIR "+PG$+" "+PL$;MZ(1);H1+".":GOSUB3500:RETURN
 3000 IFAW(3)+AW(4)=2ANDMT>5ANDPG<PF/10GOSUB3700:PRINT@517,"VICI
HAS BEEN SUCCESSFULLY INVADED!  YOU ARE OVERTHROWN":FORX=1TO15:G
OSUB3800:NEXT:CLS:END:ELSERETURN
3100 CLS:FORXO=1TO150-(100*AW(3+MB)):A=RND(1000):PRINT@A,A$:NEXT
XO:CLS:PRINTCHR$(23):PRINT@462,HD(MB+4)+RP$:GOSUB3800:GOSUB3700:
RETURN
```

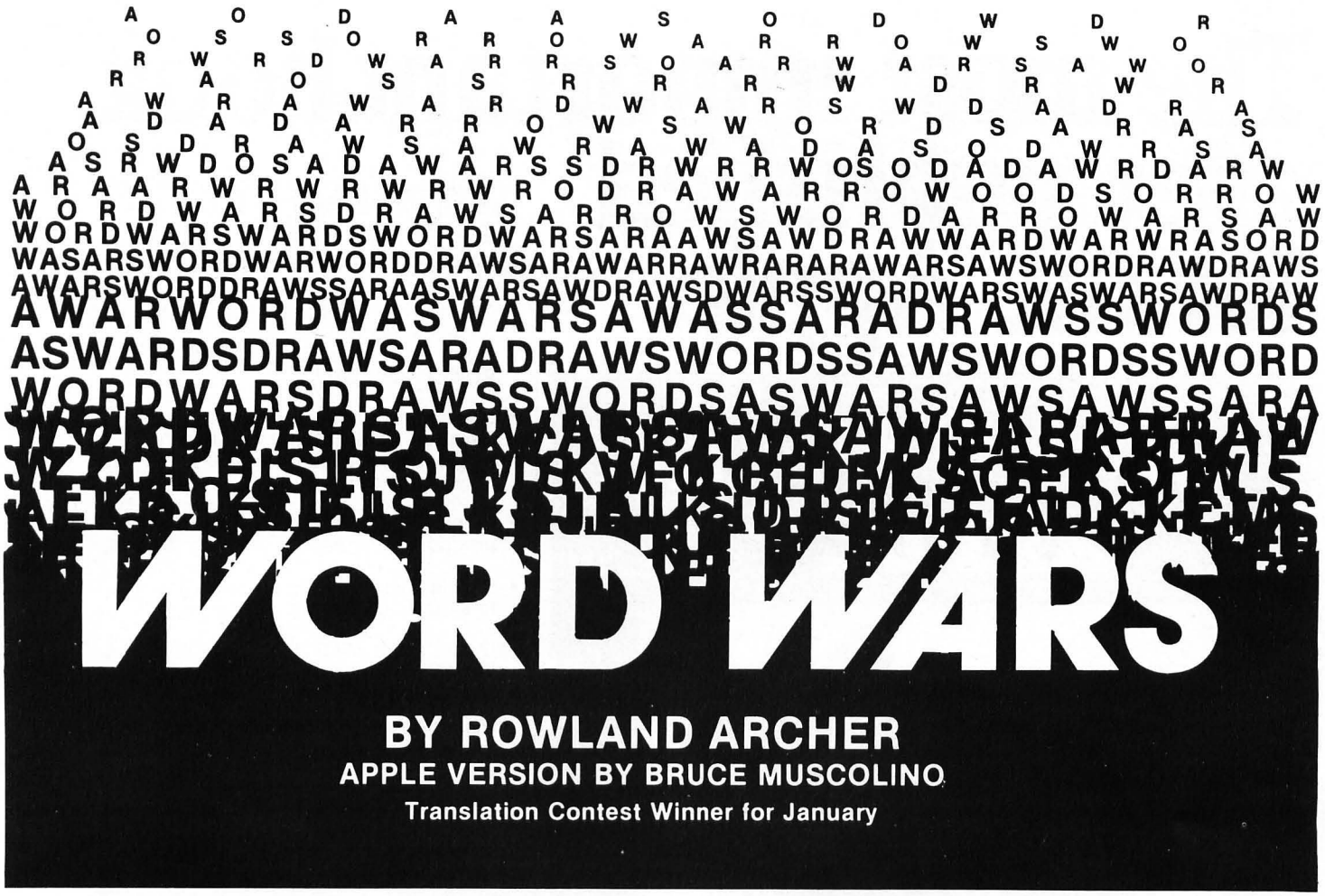
Rebellion.

```
3200 CLS:HD="  REBELLION":GOSUB3900:GOSUB3220:XL=0:A=92:CLS:GOSU
B4100:GOSUB4500:GOSUB4200:GOSUB4200:GOSUB4100:A=46:XL=13:FORX=1T
O14:GOSUB4100:GOSUB3220:XL=XL-1:NEXT
 3210 CLS:HD="YOU ARE DEPOSED!":PU=479:GOSUB3600:FORX=1TO4:GOSUB3
800:NEXT:CLS:END
```

```
3220 FORXO=1TO3:NEXTXO:RETURN
3300 CLS:PRINTCHR$(23):HD="VICIVICTORIOUS":FORMV=1TO5:AO=2:AP=28
:FORX=1TO13:Y=MID$(HD,X,1):PRINT"[";TAB(AO)Y;TAB(AP)Y;TAB(30)"["
:AO=AO+1:AP=AP-1:NEXTX:PRINT"[";TAB(AO)"S";TAB(30)"[":NEXT:PRINT
:FORX=1TO31:PRINT"+";:NEXT
3310 GOSUB3800:CLS:XL=0:A=86:GOSUB4100:GOSUB4500:FORX=0TO5:GOSUB
4200:NEXT:FORX=1TO91:READAO,AP:POKEAO,AP:NEXT:GOSUB4100:HD="ANNO
"+STR$(MY):PRINT@95-(LEN(HD)/2),HD;
3320 PRINT@909,"LONG MAY YOU RULE, SOVEREIGN OF CESSA!":FORX=1TO
10:GOSUB3800:NEXT:CLS:END
3400 IN=INKEY$:IFNOTIN="Y"ANDNOTIN="N"GOTO3400
3410 RETURN
3500 PRINT@900,"PLEASE INDICATE WHEN YOU WISH TO CONTINUE ";+YH
3510 IN=INKEY$:IFIN=""GOTO3510
3520 RETURN
3600 PS=LEN(HD):PT=PU-PS:FORX=1TOPS:A$=MID$(HD,X,1):PRINT@PT,A$:
PT=PT+2:NEXT:RETURN
3700 CLS:XL=2:A=42:GOSUB4100:XL=4:A=61:GOSUB4100:XL=11:GOSUB4100
:XL=13:A=42:GOSUB4100:PRINT@384,YH+",":RETURN
3800 FORXO=0TO200:NEXTXO:RETURN
3900 PRINTCHR$(23):PRINT@462,HD:GOSUB3800:GOSUB3800:RETURN
4000 CLS:PRINTCHR$(23):PRINT@344,TH$:PRINT@462,HD:PRINT@594,"IS
CLOSED":HD="":GOSUB3800:RETURN
4100 A$=STRING$(64,CHR$(A)):PRINT@XL*64,A$:A$="":RETURN
4200 POKE15424+(XL*64),A:POKE15487+(XL*64),A:XL=XL+1:RETURN
4300 CLS:XL=0:A=188:GOSUB4100:A=191:GOSUB4500:XK=15895:FORX=1TO4
:POKEXK,136:POKEXK+1,157:IFFR=0POKEXK+2,47
4310 XK=XK+4:NEXT:GOSUB4200:XK=15960:FORX=1TO4:POKEXK,129:XK=XK+
4:NEXT:IFAW(7)>1THENAO=143ELSEAO=191
4320 GOSUB4200:FORX=0TO47STEP47:POKE16006+X,189:POKE16007+X,189:
POKE16008+X,AO:POKE16009+X,190:POKE16010+X,190:NEXT:FORX=1TO4:GO
SUB4200:NEXT:A=143:GOSUB4100:GOSUB3500:RETURN
4400 CLS:XL=0:A=86:GOSUB4100:GOSUB4500:FORX=1TO3:GOSUB4200:NEXT:
GOSUB4100:RETURN
4500 GOSUB4200:POKE15535,168:POKE15536,42:POKE15537,148:GOSUB420
0:POKE15567,46:POKE15569,43:POKE15571,46:POKE15600,191:GOSUB4200
:POKE15631,159:POKE15632,140:POKE15633,143:POKE15634,140:POKE156
35,175
4510 POKE15663,140:POKE15664,179:POKE15665,140:GOSUB4200:POKE157
28,191:GOSUB4200:POKE15792,191:GOSUB4200:RETURN
4600 PRINT@XL*64,H(4),,:PRINTUSINGPU$;S(4);:PRINT" "+H1(4):XL=XL
+1:RETURN
```

Data statements for victory graphic.

```
4700 DATA15773,60,15774,149,15828,168,15829,95,15830,188,15831,9
5,15832,188,15833,95,15834,188,15835,95,15836,188,15837,95,15838
,149,15892,170,15893,191,15894,191,15895,191,15896,191,15897,191
,15898,191,15899,191,15900,191,15901,191,15902,149
4710 DATA15956,170,15957,43,15958,143,15960,143,15961,43,15962,1
43,15964,143,15965,43,15966,191,15967,189,15968,189,15969,189,15
970,189,15971,189,15972,189,15973,189,15974,189,15975,189,15976,
149
4720 DATA16020,170,16021,191,16022,179,16023,191,16024,179,16025
,191,16026,179,16027,191,16028,179,16029,191,16030,191,16031,190
,16032,191,16033,190,16034,191,16035,190,16036,191,16037,190,160
38,190,16039,191,16040,149
4730 DATA16084,138,16085,141,16086,143,16087,142,16088,141,16089
,143,16090,142,16091,141,16092,143,16093,142,16094,143,16095,143
,16096,143,16097,35,16098,35,16099,35,16100,143,16101,143,16102,
143,16103,143,16104,133
4740 DATA15427,42,15484,42,15747,42,15804,42,16067,42,16124,42
```



WORD WARS

BY ROWLAND ARCHER

APPLE VERSION BY BRUCE MUSCOLINO

Translation Contest Winner for January

***Word Wars* is a word game for an Apple with Applesoft and 16K RAM.**

Apple owners now have the opportunity to join the vocabulary battle, with this translation of Rowland Archer's original program which was published last July. The object is simple enough: to form as many words as possible, in a limited time, from a string of letters generated by the computer. One person may play alone, or as many as six individuals (or even groups) may compete against one another.

From the jumble of 13 letters displayed, you must type in as many three- through eight-letter words as you can find in about three minutes. However, a maximum of five words of each length is allowed, and you'll undoubtedly find yourself pushing your limit on the shorter words much more quickly than the longer ones. Scoring is based on word length.

At the end of a round, all players' words are subject to challenge by the other players. If an invalid word is found (a dictionary should be used for arbitration), a substantial penalty is given. Five rounds constitute a tournament, so that if you really bomb out on one set of letters, you can always hope for a comeback in a subsequent round.

Complete instructions for play and information on scoring are given in the program upon request (lines 1500-1940). Enjoy this challenging game of vocabulary and quick-thinking skills.

Variables

A, A$: Input character.
AD: Increment added to clock during input.
BN, BN(n): Bonus points added to each word.
BP: Bonus points.
BW: Bonus flag.
BZ: Buzzer.
CL: Clock.
CP: Current player.
FS(n): Final scores.
GN: Game number.
GW$: Letters for roll.
H: Cursor horizontal position.
I, J, J1, J2, K: General loop counters.
L: ASCII value of letters.
L$: Chosen letter in roll, and letter used in validation.
LW: Length of word.
M: Random number used to select letters in roll.
M$: Holds various messages to be printed.
NL: Number of letters.
NM$(n): Players' names.
NP: Number of players.
NS$(n): Players' names in possessive form.
NW(n): Number of word.
OS(n): Tab stops for printing words.
PC$, PD$: Dummy characters.
PN: Point reduction due to error.
QP: "Q" flag.
RL$: Final roll of letters.
S: Address to PEEK/POKE for speaker click.
SB: Score to beat.
SC, SC(n): Scores.
T, T$: Temporary variables.
TI: Ticks of the clock.
TN$(n): Players' names (temporary).
TS(n): Temporary scores.
TS$: Used in print format.
V: Cursor vertical position.
W$: Player's word.
WA(n): Array holding count of letters in roll.
WD$(n): Words formed by player in current game.
WW(n): Array used in word validation.
X: Delay loop counter.

The most important book ever published for the Apple.

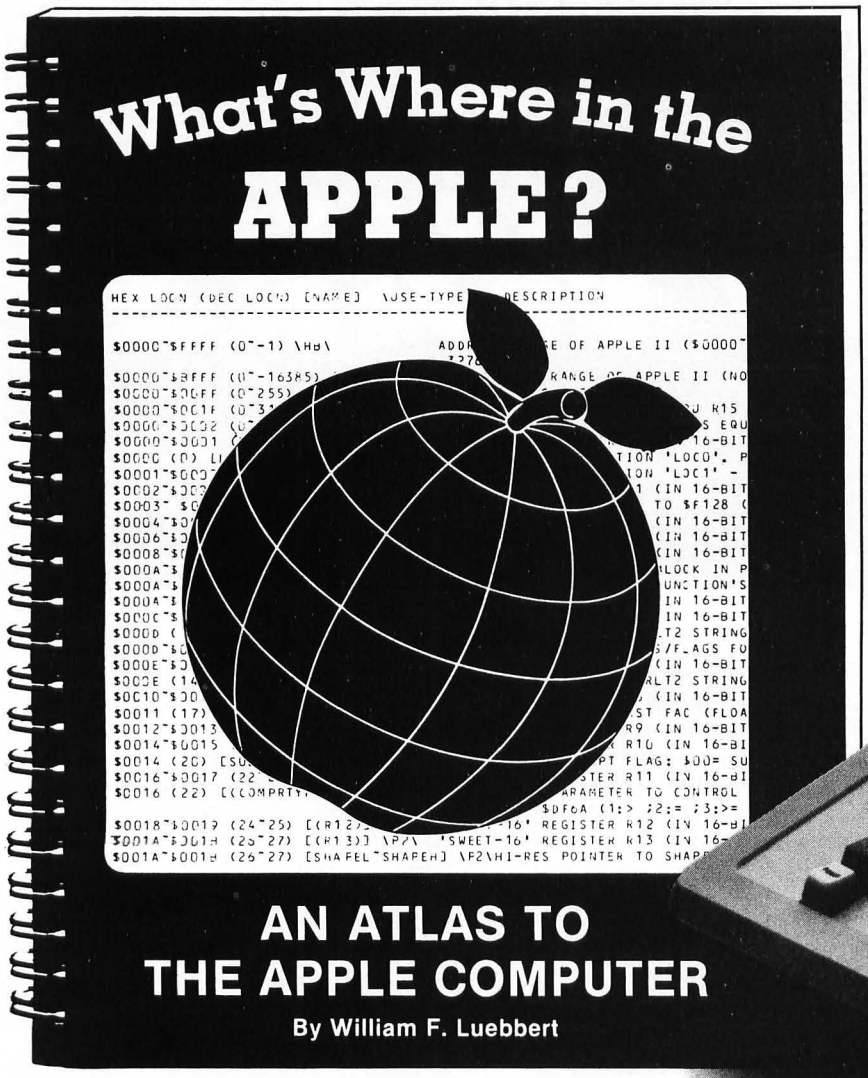
The most comprehensive description of Apple II firmware and hardware ever published — all in one place.

What's Where in the Apple?

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

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

Ask for it at your computer store



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

$14.95

ISBN: 0-938222-07-4

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

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

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

☐ VISA ☐ MasterCard

Acct. # _____ Expires _____

Name _____

Address _____

City _____ State _____ Zip _____

Signature _____

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

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$         'WORD WARS'          $
$    AUTHOR: ROWLAND ARCHER    $
$   TRANSL: BRUCE MUSCOLINO    $
$      (C) 1982    SOFTSIDE    $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
10  HOME :V = 12:H = 10: GOSUB 10
    0: PRINT "PRESS ANY KEY TO S
    TART"
15  POKE  - 16368,0
20  IF  PEEK ( - 16384) < 128 OR
     PEEK ( - 16384) > 255 THEN
    I =  RND (1): GOTO 20
25  POKE  - 16368,0
90 S =  - 16336: GOTO 500
```

Subroutine to position cursor.

```
100  HTAB H: VTAB V: RETURN
```

Delay loop subroutine.

```
105  FOR X = 1 TO 500: NEXT X: RETURN
```

Subroutine to print board verticals.

```
110  FOR V = 15 TO 20: GOSUB 100:
      INVERSE : PRINT " ": NEXT V
    : NORMAL : PRINT : RETURN
```

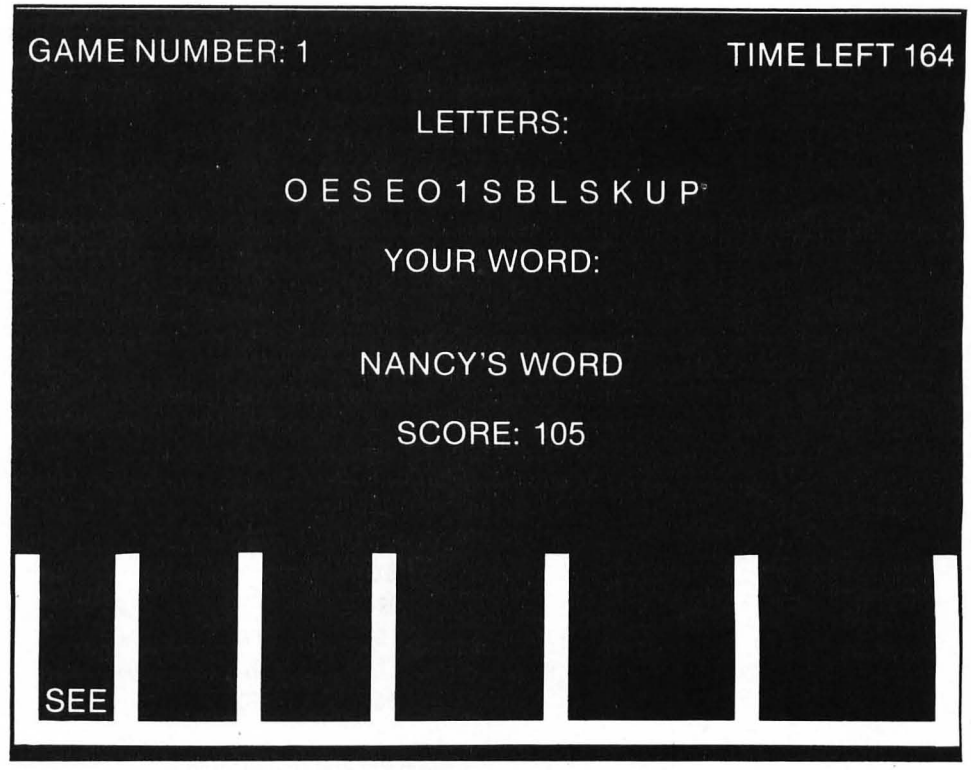
Subroutine to erase a line.

```
115  FOR H = 1 TO 40: GOSUB 100: PRINT
     " ": NEXT H: RETURN
```

Decrement clock, print time remaining; check for 60-second warning, and buzz speaker if it's time.

```
120 CL = CL - 1:TI = 0:V = 1:H =
    37: GOSUB 100: PRINT "   ": GOSUB
    100: PRINT CL
125  IF CL = 60 THEN V = 9:H = 10
    : FOR I = 1 TO 50: GOSUB 135
    : NEXT I: GOSUB 100: PRINT "
    --60 SECOND WARNING --": GOSUB
    105: GOSUB 115:TI = 30
130 V = 7:H = 20 +  LEN (W$): RETURN
```

Subroutine to buzz speaker.




```
135 BZ =  PEEK (S) +  PEEK (S): RETURN
```

Print the message contained in M$ on the screen, pause, and erase it.

```
140 V = 9: GOSUB 100: PRINT M$: GOSUB
    105: GOSUB 105: GOSUB 115:V =
    7: GOSUB 115: RETURN
```

Subroutine to draw the board on the screen.

```
145 H = 1: GOSUB 110:H = 5: GOSUB
    110:H = 10: GOSUB 110:H = 16
    : GOSUB 110:H = 23: GOSUB 11
    0:H = 31: GOSUB 110:H = 40: GOSUB
    110
150 V = 21: INVERSE : FOR H = 1 TO
    40: GOSUB 100: PRINT " ": NEXT
    H: NORMAL : RETURN
```

Subroutine to display a message to clear the area.

```
155 V = 7:H = 4: GOSUB 100: PRINT
    "IT'S ";NS$(CP);" TURN...CLE
    AR THE AREA!"
160 V = 23:H = 8: GOSUB 100: PRINT
    "PRESS ";: FLASH : PRINT "<R
    ETURN>";: NORMAL : PRINT " W
    HEN READY": RETURN
165 H = 7:V = 23: GOSUB 100: PRINT
    "PRESS <RETURN> TO CONTINUE"
```

```
170  POKE  - 16368,0
175  IF  PEEK ( - 16384) <  > 141
     THEN 175
180  POKE  - 16368,0: RETURN
```

Pause for the player to press RETURN.

```
185 V = 7:H = 8: GOSUB 100: PRINT
    "OK, ";NM$(CP);" PRESS ";: FLASH
    : PRINT "<RETURN>";: NORMAL
    : PRINT : RETURN
```

Generate the roll of letters from which to make words. If roll contains a Q, be sure it contains a U also (give the guy a break!).

```
200 QP = 0:L$ = "":RL$ = "": FOR
    I = 0 TO 25:WA(I) = 0:WW(I) =
    0: NEXT I
205  FOR I = 1 TO NL:M =  INT ( LEN
    (GW$) *  RND (1) + 1): IF M =
     LEN (GW$) THEN M = M - 1
210 L$ =  MID$ (GW$,M,1):L =  ASC
    (L$) - 65:RL$ = RL$ + L$:WA(
    L) = WA(L) + 1
215  IF L$ = "Q" THEN QP = 1
220  NEXT I
225  IF QP = 0 THEN 245
230 I =  INT (NL *  RND (1) + 1):
     IF I < 2 THEN I = 2
235  IF I > 12 THEN I = 12
240 L$ =  MID$ (RL$,I,1):L =  ASC
```

```
    (L$) - 65:WA(L) = WA(L) - 1:
    WA(20) = WA(20) + 1:RL$ =  LEFT$
    (RL$,I - 1) + "U" +  RIGHT$
    (RL$,NL - I)
245 RETURN
```

Subroutine to print scores on the screen.

```
255 FOR I = 0 TO NP - 2
260 FOR J = I + 1 TO NP - 1
265 IF TS(I) >  = TS(J) THEN 285

270 T = TS(I):T$ = TN$(I)
275 TS(I) = TS(J):TN$(I) = TN$(J)
280 TS(J) = T:TN$(J) = T$
285 NEXT J
290 NEXT I
295 V = 5:H = 12: GOSUB 100: PRINT
    "NAME":H = 24: GOSUB 100: PRINT
    "SCORE":V = 7
300 FOR I = 0 TO NP - 1
305 H = 12: GOSUB 100: PRINT TN$(
    I):TS$(I) =  STR$ (TS(I)):H =
    29 -  LEN (TS$(I)): GOSUB 10
    0: PRINT TS(I):TN$(I) = "":T
    S(I) = 0:V = V + 1
310 NEXT I
315 RETURN
```

Principal input routine: Accepts and validates letters, and recognizes special characters.

```
385 V = 7:H = 20: POKE  - 16368,0
390 TI = TI + 1: IF TI > 180 THEN
     GOSUB 120
395 IF CL = 0 THEN  GOSUB 105: FOR
    J = 1 TO 50: GOSUB 135: NEXT
    J: GOTO 1000
400 GOSUB 100: FLASH : PRINT " "
    : NORMAL :A =  PEEK ( - 1638
    4)
405 IF A =  < 128 THEN TI = TI +
    AD: GOTO 390
410 IF A = 141 THEN  POKE  - 163
    68,0: GOTO 455
420 IF A = 136 THEN  POKE  - 163
    68,0: GOTO 465
430 IF A = 155 THEN  POKE  - 163
    68,0: GOTO 480
440 IF A = 152 THEN  POKE  - 163
    68,0: GOTO 490
445 IF A >  = 193 AND A <  = 218
     THEN A = A - 128:A$ =  CHR$
    (A): GOSUB 100: PRINT A$: POKE
     - 16368,0:H = H + 1:TI = TI
     + AD:W$ = W$ + A$
450 GOTO 390
455 IF  LEN (W$) > 0 THEN 795
460 GOTO 390
465 IF H = 20 THEN 390
470 IF  LEN (W$) = 1 THEN W$ = "
    ": GOSUB 100: PRINT " ":H =
    H - 1: GOSUB 100: GOTO 390
475 W$ =  LEFT$ (W$, LEN (W$) - 1
    ): GOSUB 100: PRINT " ":H =
    H - 1: GOSUB 100: GOTO 390
480 IF NP = 1 THEN M$ = "** ABAN
    DONING THIS GAME **":H = 7: GOSUB
    140: POKE 34,0: HOME : GOTO
    1365
485 GOTO 390
490 FOR H = 20 TO 20 +  LEN (W$)
    : GOSUB 100: PRINT " ": NEXT
    H:TI = TI + AD:H = 20: GOSUB
    100:W$ = "": GOTO 390
```

Title page and initialization.

```
500 HOME :V = 5:H = 16: GOSUB 10
    0: PRINT "WORD WARS"
505 V = 9:H = 10: GOSUB 100: PRINT
    "ORIGINAL TRS-80 PROGRAM"
510 V = 10:H = 19: GOSUB 100: PRINT
    "BY"
515 V = 11:H = 13: GOSUB 100: PRINT
    "ROWLAND ARCHER"
520 V = 14:H = 14: GOSUB 100: PRINT
    "APPLE VERSION"
525 V = 15:H = 19: GOSUB 100: PRINT
    "BY"
530 V = 16:H = 13: GOSUB 100: PRINT
    "BRUCE MUSCOLINO"
535 GOSUB 105
540 V = 21:H = 14: GOSUB 100: PRINT
    "INSTRUCTIONS"
545 V = 23:H = 12: GOSUB 100: PRINT
    "PRESS <Y> OR <N>"
550 POKE  - 16368,0
555 IF  PEEK ( - 16384) <  > 217
     THEN 565
560 POKE  - 16368,0: GOTO 1500
565 IF  PEEK ( - 16384) <  > 206
     THEN 555
570 POKE  - 16368,0: HOME :H = 1
    :V = 3
575 GOSUB 100: INPUT "HOW MANY P
    EOPLE ARE PLAYING WORD WARS?
    ";NP
580 IF NP < 1 THEN V = V + 2:H =
    10: GOSUB 100: PRINT "THAT'S
     NOT VERY MANY!":V = V + 1:H
     = 7: GOSUB 100: PRINT "TRY
    AND ROUND UP SOME MORE!":V =
    V + 2:H = 1: GOSUB 105: GOTO
    575
585 IF NP > 6 THEN V = V + 2:H =
    7: GOSUB 100: PRINT "I THINK
     I'D GET LOST TRYING":V = V +
    1:H = 6: GOSUB 100: PRINT "T
    O REMEMBER ALL THOSE WORDS..
    .":V = V + 1:H = 4: GOSUB 10
    0: PRINT "LET'S KEEP IT DOWN
     TO SIX PLAYERS.": GOSUB 105
    :V = V + 2:H = 1: GOTO 575
590 DIM WD$(NP - 1,34)
595 DIM NW(NP - 1,6),NM$(NP - 1)
    ,NW$(NP - 1),TN$(NP - 1),FS(
    NP - 1),SC(NP - 1),TS(NP - 1
    ),BN(6),OS(6),WA(25),WW(25):
    NL = 13
600 HOME :V = 3:H = 10: FOR I =
    1 TO NP
605 GOSUB 100: PRINT "WHAT'S PLA
    YER #"; STR$ (I);"'S NAME":H
     = 18:V = V + 1: GOSUB 100: INPUT
    NM$(I - 1):V = V + 2:H = 10
610 NEXT I
615 FOR I = 0 TO NP - 1: IF  RIGHT$
    (NM$(I),1) = "S" THEN NS$(I)
     = NM$(I) + "'": GOTO 625
620 NS$(I) = NM$(I) + "'S"
625 NEXT I
630 PC$ = "********":PD$ = "#####
    ###"
635 GW$ = "AAAAAAAABBCCDDDDEEEEEE
    EEEEEEFFGGGHHIIIIIIIIIJKLLLL
    MMNNNNNNOOOOOOOOPPQRRRRRRSSS
    STTTTTTUUUUVVWWXYYZ"
645 OS(0) = 2:OS(1) = 6:OS(2) = 1
    1:OS(3) = 17:OS(4) = 24:OS(5
    ) = 32
650 BN(0) = 0:BN(1) = 5: FOR I =
    2 TO 6:BN(I) = BN(I - 1) * 2
    : NEXT I
655 AD = 7:GN = 1
660 BP = 1750
```

Draw screen template.

```
700 HOME :V = 1:H = 1: GOSUB 100
    : PRINT "GAME NUMBER: ";GN
705 V = 1:H = 26: GOSUB 100: PRINT
    "TIME LEFT: "
710 GOSUB 145
```

Beginning of main game loop. Generate the roll of letters for this game and initialize necessary variables.

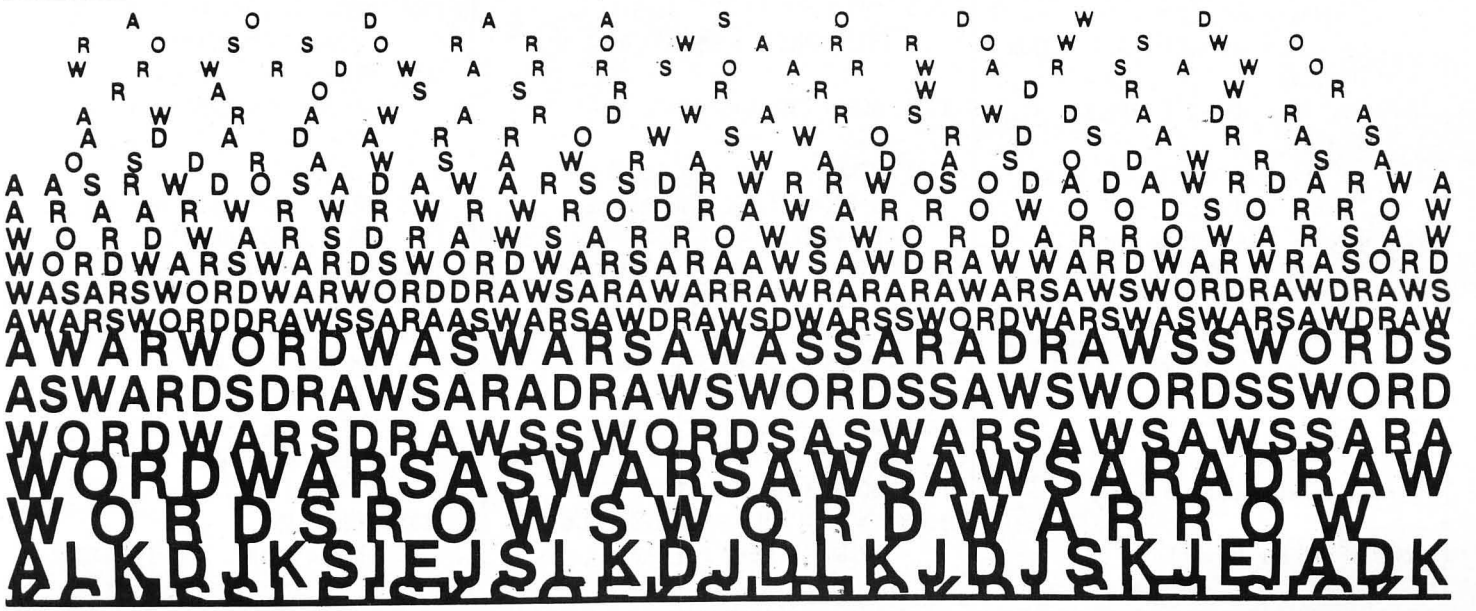
```
715  GOSUB 200:SB = 0
720  FOR CP = 0 TO NP - 1
725 CL = 180:V = 1:H = 37: GOSUB
     100: PRINT CL: IF NP = 1 THEN
      GOSUB 185: POKE 34,24: GOSUB
     170: GOTO 735
730  GOSUB 155: POKE 34,24: GOSUB
     170
735 SC(CP) = 0
740 V = 7: GOSUB 115:V = 23: GOSUB
     115
745 V = 3:H = 16: GOSUB 100: PRINT
     "LETTERS:"
750 H = 7: IF GN = 5 THEN H = 1
755 V = 5: GOSUB 100: FOR I = 1 TO
     NL: PRINT  MID$ (RL$,I,1);"
     ";: NEXT I
760 V = 11:H = 17 -  LEN (NS$(CP)
     ) / 2: GOSUB 100: PRINT NS$(
     CP);" WORDS"
765 V = 12:H = 16: GOSUB 100: PRINT
     "SCORE: "
770  IF NP > 1 THEN V = 23:H = 10
     : GOSUB 100: PRINT "SCORE TO
      BEAT: ";SB
775 BW = 0
```

Get a word using the input routine; verify that it has not been played before, has at least three letters, and contains only letters in the roll. Also check number of words of that length.

```
780  FOR I = 0 TO 25:WW(I) = WA(I
     ): NEXT I
785 V = 7:H = 9: GOSUB 100: PRINT
     "YOUR WORD:"
790 W$ = "": GOTO 385
795 V = 9:H = 15: GOSUB 100: PRINT
     "CHECKING..."
800  FOR I = 1 TO  LEN (W$):L$ =
      MID$ (W$,I,1):L =  ASC (L$)
      - 65
805  IF WA(L) = 0 THEN M$ = "** T
     HERE ARE NO " + L$ + "'S IN
     THIS ROLL **":H = 3: GOSUB 1
     40: GOTO 780
810  IF WW(L) = 0 THEN M$ = "** Y
     OUR WORD CONTAINS TOO MANY "
      + L$ + "'S **":H = 3: GOSUB
     140: GOTO 780
815 WW(L) = WW(L) - 1
820  NEXT I
825 TI = 30:LW =  LEN (W$): IF LW
      > 8 THEN LW = 8
830  IF LW < 3 THEN M$ = "** WORD
     S MUST HAVE AT LEAST 3 LETTE
     RS **":H = 1: GOSUB 140: GOTO
     780
835  IF NW(CP,LW - 3) = 5 THEN M$
      = "** YOU HAVE 5 WORDS OF T
     HAT LENGTH **":H = 2: GOSUB
     140: GOTO 780
840 NW = NW(CP,LW - 3)
845  IF NW = 0 THEN 865
850  FOR J = 5 * (LW - 3) TO 5 *
     (LW - 3) + NW - 1
855  IF WD$(CP,J) <  > W$ THEN  NEXT
     J: GOTO 865
860 M$ = "** SORRY, YOU HAVE THAT
      WORD **":H = 5: GOSUB 140: GOTO
     780
865 NW = NW(CP,LW - 3):NW(CP,LW -
     3) = NW + 1
870 WD$(CP,5 * (LW - 3) + NW) = W
     $
875  IF  LEN (W$) > 8 THEN W$ =  LEFT$
     (W$,8):LW = 8
880 SC = LW * 10
885 H = OS(LW - 3):V = 20 - NW: GOSUB
     100: PRINT W$:V = 9: GOSUB 1
     15:V = 7: GOSUB 115
890 BN = BN(LW - 3) * (NW + 1)
895  IF SC(CP) > BP THEN BN = BN +
     BN
900 SC = SC + BN:SC(CP) = SC(CP) +
     SC:V = 12:H = 23: GOSUB 100:
      PRINT "      ": GOSUB 100: PRINT
     SC(CP)
905  IF SC(CP) > BP AND BW = 0 THEN
     M$ = "** YOU'RE IN DOUBLE BO
     NUS TIME **":H = 4: GOSUB 14
     0:M$ = "** ALL BONUS SCORES
     NOW DOUBLED **":H = 3: GOSUB
     140:M$ = "** YOU GET 60 SECO
     NDS OF EXTRA TIME **":H = 1:
      GOSUB 140:CL = CL + 60:BW =
      - 1
910  GOTO 780
1000  POKE 34,2: HOME
1005  IF CP = NP - 1 THEN  POKE 3
     4,0: GOTO 1100
1010 CP = CP + 1: GOSUB 155:CP =
     CP - 1: GOSUB 145
1015  IF SC(CP) > SB THEN SB = SC
     (CP)
1020  NEXT CP
1025  IF SC(CP) > SB THEN SB = SC
     (CP)
1035  NEXT CP
```

List player's words on the screen for possible challenge by other players.

```
1100  FOR I = 0 TO NP - 1
```



```
1105  HOME
1110 V = 2:H = 17 -  INT ( LEN (N
     S$(I)) / 2): GOSUB 100: PRINT
     NS$(I);" WORDS"
1115 K = 0:V = 4:J1 = 0:J2 = 4
1120  FOR J = J1 TO J2:H = OS(K)
1125  IF WD$(I,J) = "" THEN WD$(I
     ,J) =  LEFT$ (PD$,K + 3)
1130  GOSUB 100: PRINT WD$(I,J):V
      = V + 1: NEXT J
1135  IF J > 29 THEN 1145
1140 K = K + 1:V = 4:J1 = J1 + 5:
     J2 = J2 + 5: GOTO 1120
1145 H = 1:V = 10: GOSUB 100: PRINT
     "ANY OF THESE WORDS MAY NOW
     BE CHALLENGED": POKE 34,8
1150 H = 7:V = 12: GOSUB 100: PRINT
     "PRESS <C> TO CHALLENGE A WO
     RD"
1155 H = 19:V = 13: GOSUB 100: PRINT
     "OR"
1160 H = 6:V = 14: GOSUB 100: PRINT
     "PRESS <RETURN> TO CONTINUE
     PLAY"
1165  POKE  - 16368,0
1170  IF  PEEK ( - 16384) <  > 14
     1 THEN 1180
1175  POKE  - 16368,0: POKE 34,0:
      GOTO 1240
1180  IF  PEEK ( - 16384) <  > 19
     5 THEN 1170
1185  POKE  - 16368,0: HOME
1190 H = 3:V = 10: GOSUB 100: INPUT
     "TYPE IN THE CHALLENGED WORD
     ";W$
1195  FOR J = 0 TO 29
1200  IF WD$(I,J) = W$ THEN 1215
1205  NEXT J
1210 H = 5:V = 12: GOSUB 100: PRINT
     "THAT'S NOT ONE OF ";NS$(I);
     " WORDS": GOTO 1235
1215 LW =  LEN (W$):WD$(I,J) =  LEFT$
     (PC$,LW)
1220 PN = LW * 10 + 2 * (BN(LW -
     3) * 5)
1225 H = 5:V = 12: GOSUB 100: PRINT
     "SORRY ";NM$(I);"..."
1230 H = 8:V = 14: GOSUB 100: PRINT
     "...THAT'LL COST YOU ";PN;"
     POINTS":SC(I) = SC(I) - PN
1235  GOSUB 105: GOSUB 105: GOSUB
     105: POKE 34,0: HOME : GOTO
     1110
1240  NEXT I
```

Printout for single-player game.

```
1245  IF NP > 1 THEN 1300
1250 FS(CP) = FS(CP) + SC(CP): HOME
1255 H = 6:V = 9: GOSUB 100: PRINT
     "WELL DONE ";NM$(0);"..."
1260 H = 6:V = 10: GOSUB 100: PRINT
     "YOU FINISHED WITH ";SC(0);"
      POINTS"
1265  GOSUB 105
1270 H = 4:V = 12: GOSUB 100: PRINT
     "YOUR TOTAL SCORE IS NOW ";F
     S(CP);" POINTS"
1275  GOSUB 105
1280  IF GN < 5 THEN 1355
1285 H = 10:V = 13: GOSUB 100: PRINT
     "...AND THAT'S FINAL!"
1290  GOSUB 105: GOTO 1365
```

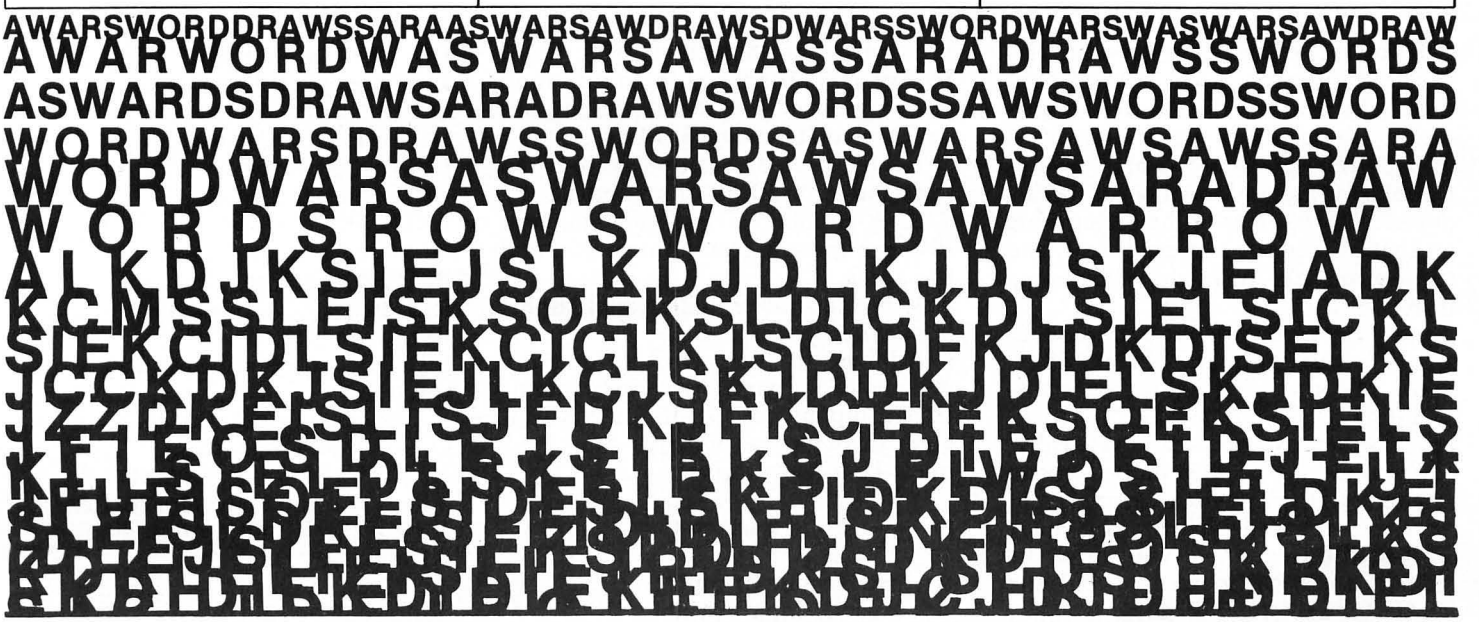
List players in descending order of their scores for the previous game, and then by their cumulative scores for the current tournament.

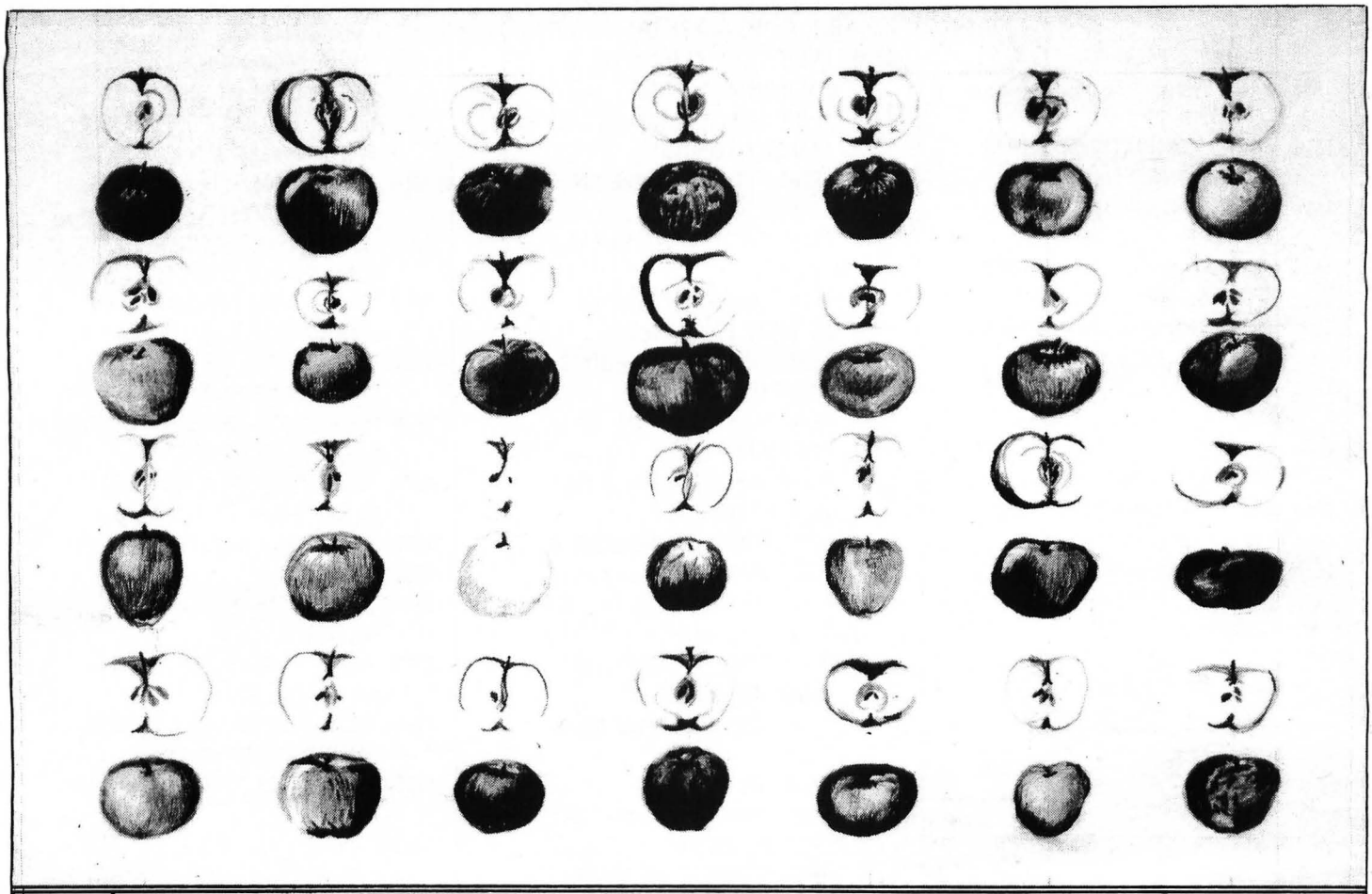
```
1300  HOME :V = 3:H = 12: GOSUB 1
     00: PRINT "SCORING FOR ROUND
     "
1305  FOR I = 0 TO NP - 1:FS(I) =
     FS(I) + SC(I):TS(I) = SC(I):
     TN$(I) = NM$(I): NEXT I
1310  GOSUB 255
1315  IF GN = 5 THEN V = 15:H = 5
     : GOSUB 100: PRINT "PRESS <R
     ETURN> FOR FINAL SCORES": GOSUB
     170: GOTO 1340
1320  IF GN < 5 THEN V = 15:H = 2
     : GOSUB 100: PRINT "PRESS <R
     ETURN> FOR CUMULATIVE SCORES
     ": GOSUB 170
1325  HOME :V = 3:H = 12: GOSUB 1
     00: PRINT "CUMULATIVE SCORES
     "
1330  FOR I = 0 TO NP - 1:TS(I) =
     FS(I):TN$(I) = NM$(I): NEXT
     I
1335  GOSUB 255: GOSUB 165: GOTO
     1355
1340  HOME :V = 3:H = 6: GOSUB 10
     0: PRINT "FINAL SCORING FOR
     TOURNAMENT"
1345  FOR I = 0 TO NP - 1:TS(I) =
     FS(I):TN$(I) = NM$(I): NEXT
     I
1350  GOSUB 255: GOSUB 165: GOTO
     1370
1355 GN = GN + 1: IF GN = 5 THEN
     NL = 20
1360  GOSUB 165: GOTO 1395
1365 V = 15:H = 1: GOSUB 100: PRINT
     "PRESS <RETURN> FOR ANOTHER
     TOURNAMENT":V = 16:H = 12: GOSUB
     100: PRINT "PRESS <Q> TO QUI
     T": POKE  - 16368,0
1370  IF  PEEK ( - 16384) <  > 14
     1 THEN 1380
1375  POKE  - 16368,0: POKE 34,0:
      HOME : GOTO 1390
1380  IF  PEEK ( - 16384) <  > 20
     9 THEN 1370
1385  POKE  - 16368,0: POKE 34,0:
      HOME :V = 12:H = 1: GOSUB 1
     00: PRINT "THANKS FOR PLAYIN
     G - HOPE YOU HAD FUN!": GOSUB
     105: GOSUB 105: HOME : END
1390  FOR I = 0 TO NP - 1:FS(I) =
     0: NEXT I:NL = 13:GN = 1
1395  FOR I = 0 TO NP - 1: FOR J =
     0 TO 34:WD$(I,J) = "": NEXT
     J: FOR K = 0 TO 6:NW(I,K) =
     0: NEXT K: NEXT I
1400  GOTO 700
```

Print instructions.

```
1500  HOME : PRINT  TAB( 14)"INST
     RUCTIONS": PRINT
1505  PRINT "THE OBJECT OF WORD W
     ARS IS TO MAKE AS"
1510  PRINT "MANY WORDS AS YOU CA
     N FROM A GROUP OF"
1515  PRINT "LETTERS DISPLAYED ON
      THE SCREEN, WHILE"
1520  PRINT "WORKING AGAINST A TI
     ME LIMIT OF ABOUT"
1525  PRINT "THREE MINUTES.": PRINT
1530  PRINT "WORD WARS MAY BE PLA
     YED BY A SINGLE"
1535  PRINT "PLAYER, COMPETING AG
     AINST THE CLOCK,"
1540  PRINT "AND TRYING TO GET TH
     E HIGHEST POSSIBLE"
1545  PRINT "SCORE, OR BY UP TO S
     IX PEOPLE PLAYING"
1550  PRINT "AGAINST THE CLOCK AN
     D EACH OTHER.": PRINT
1555  PRINT "YOUR WORDS MUST CONT
     AIN AT LEAST THREE"
1560  PRINT "LETTERS.  THEY ARE G
     ROUPED ON THE"
1565  PRINT "SCREEN ACCORDING TO
     LENGTH.  YOU CAN"
1570  PRINT "MAKE UP TO FIVE WORD
     S OF EACH LENGTH"
1575  PRINT "FROM THREE TO EIGHT
     LETTERS."
1585  GOSUB 165
1600  HOME : PRINT  TAB( 14)"INST
```

```
    RUCTIONS": PRINT
1605  PRINT "NO PROPER NOUNS ARE
    ALLOWED, BUT"
1610  PRINT "CONTRACTIONS, PLURAL
    S, AND ALL PREFIXES"
1615  PRINT "AND SUFFIXES ARE FIN
    E.": PRINT
1620  PRINT "A GROUP OF FIVE GAME
    S IS CALLED A"
1625  PRINT "TOURNAMENT.  IN THE
    FIRST FOUR GAMES,"
1630  PRINT "YOU MUST MAKE YOUR W
    ORDS FROM A ROLL OF"
1635  PRINT "13 LETTERS.  IN THE
    LAST GAME OF THE"
1640  PRINT "TOURNAMENT YOU WILL
    HAVE 20 LETTERS TO"
1645  PRINT "WORK WITH.": PRINT
1650  PRINT "IF THERE IS MORE THA
    N ONE PLAYER, EACH"
1655  PRINT "GETS A THREE MINUTE
    TURN WITH THE SAME"
1660  PRINT "SET OF LETTERS.  NAT
    URALLY, THOSE WHO"
1665  PRINT "HAVEN'T HAD THEIR TU
    RN YET SHOULD NOT"
1670  PRINT "WATCH THE SCREEN WHI
    LE OTHERS ARE"
1675  PRINT "MAKING THEIR WORDS."
1680  GOSUB 165
1700  HOME : PRINT  TAB( 14)"INST
    RUCTIONS": PRINT
1705  PRINT "AFTER EACH PLAYER HA
    S HAD A TURN,"
1710  PRINT "EVERYONE'S WORDS ARE
     DISPLAYED IN TURN"
1715  PRINT "FOR POSSIBLE CHALLEN
    GES FROM THE OTHER"
1720  PRINT "PLAYERS.  YOU WILL N
    EED A DICTIONARY OR"
1725  PRINT "SOME OTHER MEANS OF
    ARBITRATION.": PRINT
1730  PRINT "ALL PLAYERS ARE THEN
     RANKED ACCORDING"
1735  PRINT "TO THEIR SCORE FOR T
    HIS GAME AND THEIR"
1740  PRINT "TOTAL SCORE SO FAR."
    : PRINT
1745  PRINT "A WORD ON SCORING.
    THE BASIC SCORE FOR"
1750  PRINT "A WORD IS 10 POINTS
    TIMES THE NUMBER OF"
1755  PRINT "LETTERS IN THE WORD.
      IN ADDITION, A"
1760  PRINT "BONUS RANGING FROM 0
     TO 800 POINTS IS"
1765  PRINT "ADDED TO THE SCORE O
    F EACH WORD.  THE"
1770  PRINT "MORE LETTERS IN A WO
    RD, AND THE MORE"
1775  PRINT "WORDS OF THAT LENGTH
     WHICH YOU HAVE,"
1780  PRINT "THE HIGHER THE BONUS
    ."
1785  GOSUB 165
1800  HOME : PRINT  TAB( 14)"INST
    RUCTIONS": PRINT
1805  PRINT "FINALLY, IF YOU GET
    MORE THAN 1750"
1810  PRINT "POINTS IN A GAME, YO
    U GO INTO DOUBLE"
1815  PRINT "BONUS MODE.  THE BON
    US FOR EACH"
1820  PRINT "ADDITIONAL WORD YOU
    MAKE IS DOUBLED."
1825  PRINT "YOU ALSO RECEIVE ANO
    THER 60 SECONDS OF"
1830  PRINT "PLAYING TIME.": PRINT
1835  PRINT "IF ANY OF YOUR WORDS
     ARE CHALLENGED AND"
1840  PRINT "FOUND INVALID, YOU A
    RE PENALIZED AT THE"
1845  PRINT "HIGHEST POSSIBLE SCO
    RE FOR A WORD OF"
1850  PRINT "THAT SIZE.  THAT IS,
     THE BASE SCORE"
1855  PRINT "PLUS THE HIGHEST POS
    SIBLE DOUBLE BONUS.": PRINT
1860  PRINT "WHEN ENTERING A WORD
    , BACK-ARROW WILL"
1865  PRINT "ERASE THE LAST LETTE
    R WHILE <CTRL-X>"
1870  PRINT "WILL ERASE THE ENTIR
    E WORD.  IF YOU ARE"
1875  PRINT "PLAYING ALONE, YOU C
    AN GET A NEW ROLL"
1880  PRINT "BY PRESSING <ESC>"
1885  GOSUB 165
1900  HOME : PRINT  TAB( 14)"INST
    RUCTIONS": PRINT
1905  PRINT "YOU MAY FIND IT FUN
    AT PARTIES TO LET"
1910  PRINT "TWO OR THREE PLAYERS
     WORK TOGETHER AS A"
1915  PRINT "TEAM.  AT ANY RATE..
    ."
1920 V = 7:H = 16: GOSUB 100: FLASH
    : PRINT "HAVE FUN": NORMAL :
     PRINT
1925 V = 9:H = 6: GOSUB 100: PRINT
    "AND IMPROVE YOUR VOCABULARY
    !"
1930 H = 7:V = 23: GOSUB 100: PRINT
    "PRESS <RETURN> TO BEGIN PLA
    Y": POKE  - 16368,0
1935  IF  PEEK ( - 16384) <  > 14
    1 THEN 1935
1940  POKE  - 16368,0: HOME : GOTO
    570
```





APPLE CAPTURE by William J. Ryan

***Apple Capture* is a Hi-Res color graphics game of deduction for two players, requiring Applesoft and 32K RAM.**

If you picked an Apple for your personal computer, then you'll undoubtedly enjoy this clever apple-pickin' game. Each player controls a worm (red or blue) which roams through an orchard looking for hidden apples. The 80 trees in the orchard are arranged in ten rows of eight trees, and the worms take turns moving among them.

On your turn, use the keyboard to move your worm in any of eight directions. The keys used are the ones surrounding the K key (U I O J L M , .), which will move the worm vertically, horizontally, or diagonally, corresponding to their physical arrangement on the keyboard. When you have reached the tree where you think the hidden apple is located, press the RETURN key. If your guess is right, your score is increased by one and the game continues with another hidden apple. The first player to find 15 apples wins.

Each time you try a tree which does not have the hidden apple, you'll receive a clue that will help you (and your opponent) the next time. You can get three different types of clues:

1. One or more arrows pointing in the general direction of an apple.
2. A number giving the minimum distance (in squares) between you and the apple.
3. A rain shower, indicating that this is not the right tree.

Adding to the Hi-Res graphics delights of the game are various sounds to accompany your stroll (or is it wiggle?) through the orchard. May your quest be a fruitful one!

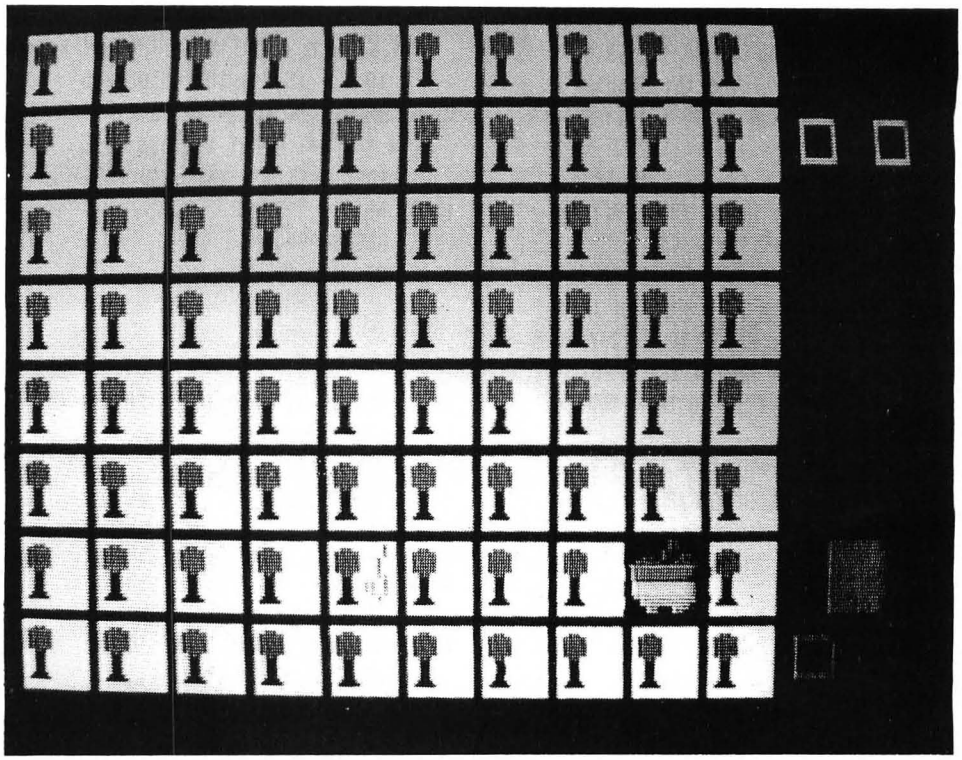
Variables

A: ASCII value of input.
A$: Answer input variable.
B: Good move flag.
C: Column.
CI: Column input.
CL: Color to plot.
DR: Direction to apple.
DS: Distance to apple.
LC(0): Apple column.
LC(1): Player 1 column.
LC(2): Player 2 column.
LR(0): Apple row.
LR(1): Player 1 row.
LR(2): Player 2 row.
N: Number to be plotted.
P: Player number.
R: Row.
RI: Row input.
RT: Direction (used to calculate rotation).
S: Segments to plot for each number.
Q, Z: General counters.
X, Y: X and Y coordinates.
XP: X coordinate to plot (offset from X by player number to produce different color).
ZZ: Type of clue (1 = direction, 2 = distance, 3 = rain).

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC         $
$       'APPLE CAPTURE'         $
$  AUTHOR: WILLIAM J. RYAN      $
$     (C) 1982     SOFTSIDE     $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Main program.

```
10  GOSUB 20000
20  HOME : PRINT "DO YOU WANT INS
    TRUCTIONS?";: GET A$
30  IF A$ = "Y" THEN  GOSUB 10000
40  GOSUB 2400
50 LR(1) = 1:LC(1) = 1:LR(2) = 10
    :LC(2) = 8: GOSUB 2600
60 P = 1:R = LR(P):C = LC(P): GOSUB
    1800: GOSUB 1200: GOSUB 2100
70 P = 2:R = LR(P):C = LC(P): GOSUB
    1800: GOSUB 1200: GOSUB 2100
80 P =  INT ( RND (1) * 2 + 1)
90  IF P = 1 THEN P = 2: GOTO 110
100 P = 1
110  GOSUB 2300
120  GOSUB 2200
130 R = LR(P):C = LC(P): GOSUB 18
    00
140  IF B = 1 THEN 200
150 CL = 0: GOSUB 1700
160 R = RI:C = CI
170 LR(P) = R:LC(P) = C: GOSUB 18
    00: GOSUB 1200
180  GOTO 120
190  IF A = 46 THEN CI = CI + 1:R
    I = RI + 1: GOTO 2270
200  GOSUB 1900: GOSUB 2000
210  IF DS = 0 THEN 330
220 ZZ =  INT ( RND (1) * 2 + 1)
230  IF  RND (1) < .1 THEN ZZ = 3
    : GOTO 250
240 CL = 3:R = LR(P):C = LC(P): GOSUB
    1800: GOSUB 1600
250  ON ZZ GOTO 260,310,320
260 N = DS: GOSUB 1800:CL = 0: GOSUB
    1300
270  FOR Z = 1 TO 1000: NEXT Z
280 R = LR(P):C = LC(P): GOSUB 18
    00
290 CL = 3: GOSUB 1600: GOSUB 110
    0: GOSUB 1200
300  GOTO 90
310 RT = DR: GOSUB 1400: GOTO 270
320  GOSUB 2500: GOTO 270
330 SC(P) = SC(P) + 1
340  GOSUB 1500
350  GOSUB 2100
360  IF SC(P) = 15 THEN 510
370 R = LR(P):C = LC(P)
380  GOSUB 1800
```



```
390  HCOLOR= 0
400  IF X > 0 THEN X = X - 2
410  IF Y > 0 THEN Y = Y - 2
420  FOR Z = 1 TO 2000: NEXT Z
430  FOR Z = 0 TO 22: HPLOT Z + X
    ,Y TO Z + X,Y + 22: NEXT Z
440 R = LR(1):C = LC(1): GOSUB 18
    00:CL = 3: GOSUB 1600: GOSUB
    1100
450 R = LR(2):C = LC(2): GOSUB 18
    00:CL = 3: GOSUB 1600: GOSUB
    1100
460 NP = P
470 LR(1) = 1:LC(1) = 1:LR(2) = 1
    0:LC(2) = 8
480 R = 1:C = 1: GOSUB 1800:P = 1
    : GOSUB 1200
490 R = 10:C = 8: GOSUB 1800:P =
    2: GOSUB 1200
500  GOSUB 2600:P = NP: GOTO 90
510  FOR Z = 1 TO 2000: NEXT Z
520  TEXT : HOME : VTAB 5: HTAB 1
    0: INVERSE : PRINT "THE WINN
    ER IS"
530  IF P = 1 THEN A$ = "BLUE"
540  IF P = 2 THEN A$ = "RED"
550  PRINT : HTAB 15: PRINT A$: PRINT
    : GOSUB 3200
560  HTAB 5: PRINT "SCORE -- ";
570  IF P = 1 THEN  PRINT SC(1);
580  IF P = 2 THEN  PRINT SC(2);
590  PRINT "  TO  ";: IF P = 1 THEN
     PRINT SC(2)
600  IF P = 2 THEN  PRINT SC(1)
```

```
610  PRINT : PRINT : GOSUB 3100
620  PRINT : PRINT "PLAY AGAIN (Y
    /N)?": GET A$: IF A$ = "Y" THEN
    40
630  PRINT : PRINT : NORMAL : PRINT
    "BYE": END
```

Draw apple tree.

```
1100  HCOLOR= 1
1140  HPLOT X + 3,Y + 2 TO X + 7,
    Y + 2: HPLOT X + 2,Y + 3 TO
    X + 8,Y + 3
1150  FOR Z = 4 TO 8: HPLOT X + 1
    ,Y + Z TO X + 9,Y + Z: NEXT
    Z
1160  HPLOT X + 3,Y + 9 TO X + 7,
    Y + 9
1165  GOSUB 2700
1170  HCOLOR= 0: FOR Z = 4 TO 6: HPLOT
    X + Z,Y + 10 TO X + Z,Y + 17
    : NEXT Z
1180  HPLOT X + 3,Y + 16 TO X + 7
    ,Y + 16: HPLOT X + 2,Y + 17 TO
    X + 8,Y + 17
1190  RETURN
```

Draw worm.

```
1200  HCOLOR= (P - 1) * 3 + 2:XP =
    X - P
1240  HPLOT XP + 18,Y + 1: HPLOT
    XP + 19,Y + 2
1250  HPLOT XP + 18,Y + 2 TO XP +
```

```
    18,Y + 4: HPLOT XP + 17,Y +
    3 TO XP + 17,Y + 16: HPLOT X
    P + 16,Y + 4 TO XP + 16,Y +
    10
1260  HPLOT XP + 18,Y + 9 TO XP +
    18,Y + 15: HPLOT XP + 19,Y +
    12 TO XP + 19,Y + 13
1265  GOSUB 2800
1270  HPLOT XP + 13,Y + 11 TO XP +
    14,Y + 12: HPLOT XP + 12,Y +
    11 TO XP + 17,Y + 16: HPLOT
    XP + 11,Y + 11 TO XP + 16,Y +
    16
1280  HPLOT XP + 11,Y + 12 TO XP +
    15,Y + 16: HPLOT XP + 11,Y +
    13 TO XP + 12,Y + 14
1290  RETURN
```

Draw number.

```
1300  HCOLOR= CL
1340  FOR Z = 1 TO 7:Q = S(N,Z): IF
    Q = 0 THEN 1395
1350  ON Q GOTO 1360,1365,1370,13
    75,1380,1385,1390
1360  HPLOT X + 3,Y + 3 TO X + 15
    ,Y + 3: HPLOT X + 4,Y + 4 TO
    X + 14,Y + 4: GOTO 1395
1365  HPLOT X + 3,Y + 4 TO X + 3,
    Y + 9: HPLOT X + 4,Y + 5 TO
    X + 4,Y + 9: GOTO 1395
1370  HPLOT X + 15,Y + 4 TO X + 1
    5,Y + 9: HPLOT X + 14,Y + 5 TO
    X + 14,Y + 9: GOTO 1395
1375  HPLOT X + 5,Y + 9 TO X + 14
    ,Y + 9: HPLOT X + 5,Y + 10 TO
    X + 14,Y + 10: GOTO 1395
1380  HPLOT X + 3,Y + 10 TO X + 4
    ,Y + 15: HPLOT X + 4,Y + 9 TO
    X + 4,Y + 14: GOTO 1395
1385  HPLOT X + 14,Y + 9 TO X + 1
    4,Y + 14: HPLOT X + 15,Y + 9
     TO X + 15,Y + 15: GOTO 1395
1390  HPLOT X + 4,Y + 14 TO X + 1
    4,Y + 14: HPLOT X + 3,Y + 15
     TO X + 15,Y + 15: GOTO 1395
1395  NEXT Z
1397  GOSUB 2900
1399  RETURN
```

Draw arrow.

```
1400  SCALE= 1: ROT= (RT - 1) * 8
    : HCOLOR= 0
1450  DRAW 1 AT X + 9,Y + 9
1452  GOSUB 2900
1460  RETURN
```

Draw apple.

```
1500  HCOLOR= 0: FOR Z = 0 TO 20:
     HPLOT X + Z,Y TO X + Z,Y +
    20: NEXT Z
1535  IF X > 0 THEN X = X - 2
1537  IF Y > 0 THEN Y = Y - 2
1540  HCOLOR= 1: HPLOT X + 14,Y: HPLOT
    X + 13,Y + 1 TO X + 15,Y + 1
    : HPLOT X + 12,Y + 2 TO X +
    15,Y + 2: HPLOT X + 12,Y + 3
     TO X + 15,Y + 3
1550  HPLOT X + 11,Y + 4 TO X + 1
    4,Y + 4: HPLOT X + 11,Y + 5 TO
    X + 13,Y + 5: HPLOT X + 11,Y
     + 6 TO X + 12,Y + 6: HPLOT
    X + 4,Y + 6 TO X + 7,Y + 6: HPLOT
    X + 16,Y + 6 TO X + 20,Y + 6
    : HPLOT X + 17,Y + 5 TO X +
    19,Y + 5
1560  HPLOT X + 2,Y + 7 TO X + 9,
    Y + 7: HPLOT X + 15,Y + 7 TO
    X + 21,Y + 7: HPLOT X + 1,Y +
    8 TO X + 21,Y + 8
1570  HCOLOR= 5: HPLOT X,Y + 9 TO
    X + 20,Y + 9: HPLOT X,Y + 10
     TO X + 20,Y + 10: HPLOT X,Y
     + 11 TO X + 20,Y + 11: HPLOT
    X,Y + 12 TO X + 19,Y + 12
1580  HCOLOR= 2: HPLOT X + 1,Y +
    13 TO X + 20,Y + 13: HPLOT X
     + 1,Y + 14 TO X + 20,Y + 14
    : HPLOT X +  + 1,Y + 15 TO X
     + 20,Y + 15: HPLOT X + 2,Y +
    16 TO X + 21,Y + 16
1590  HCOLOR= 6: HPLOT X + 2,Y +
    17 TO X + 21,Y + 17: HPLOT X
     + 3,Y + 18 TO X + 20,Y + 18
    : HPLOT X + 4,Y + 19 TO X +
    19,Y + 19
1595  HPLOT X + 5,Y + 20 TO X + 9
    ,Y + 20: HPLOT X + 13,Y + 20
     TO X + 18,Y + 20: HPLOT X +
    6,Y + 21 TO X + 8,Y + 21: HPLOT
    X + 14,Y + 21 TO X + 17,Y +
    21
1597 RP = 20: GOSUB 3000
1599  RETURN
```

Draw white square.

```
1600  HCOLOR= CL
1640  FOR Z = 0 TO 20: IF X + Z >
    279 THEN 1660
1650  HPLOT X + Z,Y TO X + Z,Y +
    20
1660  NEXT Z
1690  RETURN
```

Set half square to white.

```
1700  HCOLOR= 3
1740  FOR Z = X + 10 TO X + 18: HPLOT
    Z,Y TO Z,Y + 18: NEXT Z
1750  RETURN
```

Convert row, column to x, y coordinates.

```
1800 X = (R - 1) * 24:Y = (C - 1)
     * 24
1840  RETURN
```

Calculate distance to apple.

```
1900 RD =  ABS (LR(P) - LR(0))
1940 CD =  ABS (LC(P) - LC(0))
1950  IF RD > CD THEN DS = RD
1960  IF CD > RD THEN DS = CD
1970  IF RD = CD THEN DS = RD
1980  RETURN
```

Get direction to apple.

```
2000 DR = 0
2035  IF LR(0) > LR(P) AND LC(0) =
     LC(P) THEN DR = 1
2040  IF LR(0) > LR(P) AND LC(0) >
     LC(P) THEN DR = 2
2045  IF LR(0) = LR(P) AND LC(0) >
     LC(P) THEN DR = 3
2050  IF LR(0) < LR(P) AND LC(0) >
     LC(P) THEN DR = 4
2055  IF LR(0) < LR(P) AND LC(0) =
     LC(P) THEN DR = 5
2060  IF LR(0) < LR(P) AND LC(0) <
     LC(P) THEN DR = 6
2065  IF LR(0) = LR(P) AND LC(0) <
     LC(P) THEN DR = 7
2070  IF LR(0) > LR(P) AND LC(0) <
     LC(P) THEN DR = 8
2090  RETURN
```

Plot player score.

```
2100 N = 0
2130  IF P = 1 THEN C = 2
2140  IF P = 2 THEN C = 8
2150 R = 11:N =  INT (SC(P) / 10)

2155  GOSUB 1800
2160 CL = 0: GOSUB 1600:CL = (P -
     1) * 3 + 2: GOSUB 1300
2165 R = 12: GOSUB 1800
2170 N = SC(P) - N * 10:CL = 0: GOSUB
     1600
2180 CL = (P - 1) * 3 + 2: GOSUB
     1300
2185  FOR Q = 1 TO 500: NEXT Q
2190  RETURN
```

Get player move.

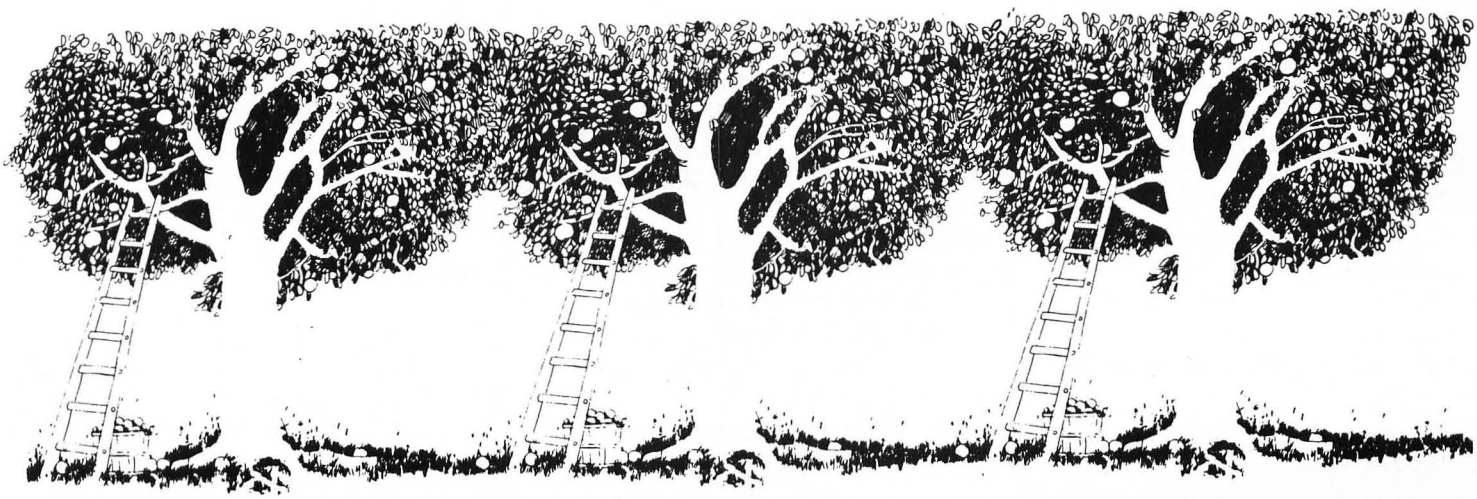
```
2200 B = 0
2230 RI = LR(P):CI = LC(P):R1 = L
     R(1 + (P < 2)):C1 = LC(1 + (
     P < 2))
2240  GET A$:A =  ASC (A$): IF A =
     13 THEN 2295
2250  IF A = 73 THEN CI = CI - 1:
      GOTO 2270
2252  IF A = 79 THEN CI = CI - 1:
     RI = RI + 1: GOTO 2270
2254  IF A = 76 THEN RI = RI + 1:
      GOTO 2270
2256  IF A = 46 THEN CI = CI + 1:
     RI = RI + 1: GOTO 2270
2258  IF A = 44 THEN CI = CI + 1:
      GOTO 2270
2260  IF A = 77 THEN CI = CI + 1:
     RI = RI - 1: GOTO 2270
2262  IF A = 74 THEN RI = RI - 1:
      GOTO 2270
2264  IF A = 85 THEN CI = CI - 1:
     RI = RI - 1: GOTO 2270
2266  FOR Z = 1 TO 30:Q =  PEEK (
      - 16336): NEXT Z
2268  GOTO 2230
2270  IF RI > 10 THEN RI = 1
2272  IF RI < 1 THEN RI = 10
2274  IF CI > 8 THEN CI = 1
2276  IF CI < 1 THEN CI = 8
2280  IF RI = R1 AND CI = C1 THEN
     2266
2285  RETURN
2295 B = 1: RETURN
```

Highlight player score.

```
2300  IF P = 2 THEN 2370
2340 R = 11.5:C = 1:CL = 2: GOSUB
     1800: GOSUB 1600
2345 C = 7:CL = 0: GOSUB 1800: GOSUB
     1600
2350  GOTO 2382
2370 R = 11.5:C = 7:CL = 5: GOSUB
     1800: GOSUB 1600
2380 C = 1:CL = 0: GOSUB 1800: GOSUB
     1600
2382  GOSUB 3100
2390  RETURN
```

Initialize board.

```
2400  HGR2 : HCOLOR= 3: HPLOT 0,0
     : CALL 62454
2435  HCOLOR= 0
2440  FOR Z = 240 TO 279: HPLOT Z
     ,0 TO Z,191: NEXT Z
2445  FOR Z = 21 TO 237 STEP 24: FOR
     Q = 0 TO 2: HPLOT Z + Q,0 TO
     Z + Q,191: NEXT Q,Z
2450  FOR Z = 21 TO 189 STEP 24: FOR
     Q = 0 TO 2: HPLOT 0,Z + Q TO
     239,Z + Q: NEXT Q,Z
2460  FOR R = 1 TO 10: FOR C = 1 TO
     8: GOSUB 1800: GOSUB 1100: NEXT
     C,R
2470 SC(1) = 0:SC(2) = 0
2490  RETURN
```



Make it rain.

```
2500  GOSUB 1800: GOSUB 1700
2540  HCOLOR= 0: HPLOT X + 15,Y +
     15 TO X + 17,Y + 15: HPLOT X
      + 14,Y + 16 TO X + 18,Y + 1
     6: HPLOT X + 15,Y + 17 TO X +
     17,Y + 17
2550  HCOLOR= 6
2560  FOR Z = 1 TO 60
2570 XA =  INT ( RND (1) * 19):YA
      =  INT ( RND (1) * 14)
2580  HPLOT X + XA,Y + YA TO X +
     XA + 1,Y + YA
2582  POKE 768, RND (1) * 50 + 40
     : POKE 769, INT ( RND (1) *
     5 + 3): CALL 770
2590  NEXT Z
2595  RETURN
```

Pick apple location.

```
2600 LR(0) =  INT ( RND (1) * 10 +
     1)
2640 LC(0) =  INT ( RND (1) * 8 +
     1)
2690  RETURN
```

Tree-planting sound.

```
2700  POKE 768,150: POKE 769,7: CALL
     770
2740  POKE 768,255: POKE 769,9: CALL
     770
2750  RETURN
```

Worm sound.

```
2800  FOR Z = 200 TO 100 STEP  -
     20
2840  POKE 768,Z: POKE 769,5: CALL
     770
2850  NEXT Z
2860  RETURN
```

Clue sound.

```
2900  POKE 768,75: POKE 769,6: CALL
     770
2940  RETURN
```

Apple sound.

```
3000  FOR Z = 1 TO RP
3040  POKE 768,193: POKE 769,8: CALL
     770
3050  POKE 768,156: POKE 769,8: CALL
     770
3060  POKE 768,128: POKE 769,8: CALL
     770
3070  POKE 768,97: POKE 769,8: CALL
     770
3080  NEXT Z
3090  RETURN
```

New turn sound.

```
3100  FOR Z = 1 TO 7
3140  POKE 768,110: POKE 769,10: CALL
     770
3150  POKE 768,130: POKE 769,10: CALL
     770
3160  NEXT Z
3170  RETURN
```

Winner sound.

```
3200 RP = 5
3240  FOR Q = 1 TO 30
3250  ON ( INT ( RND (1) * 5 + 1)
     ) GOSUB 2700,2800,2900,3000,
     3100
3260  NEXT Q
3270  RETURN
```

Print instructions.

```
10000  HOME : PRINT "  OBJECT OF
     THE GAME IS TO FIND 15": PRINT
     "APPLES IN THE ORCHARD."
10050  PRINT "  I WILL PLANT AN O
     RCHARD OF 80 TREES": PRINT "
     10 ROWS BY 8 COLUMNS. EACH P
     LAYER WILL": PRINT "HAVE A W
     ORM. ONE PLAYER WILL BE RED"
     : PRINT "AND THE OTHER BLUE.
     "
10060  PRINT "  WHEN YOUR COLOR I
     S LIT YOU MAY MOVE": PRINT "
     YOUR WORM TO ANY TREE USING
     THE ": PRINT "'U I O J L M ,
      .' KEYS. WHEN YOU THINK"
10070  PRINT "YOU HAVE FOUND THE
     APPLE, PRESS": PRINT "'RETUR
     N'. IF IT IS THE APPLE YOU G
     ET A": PRINT "POINT. IF NOT
     YOU WILL BE GIVEN 1 OF 3": PRINT
     "POSSIBLE CLUES."
10080  PRINT : PRINT "1. DIRECTIO
     N-GENERAL DIRECTION TO APPLE
     ": PRINT "2. DISTANCE-MINUMU
     N DISTANCE TO APPLE": PRINT
     "3. RAIN-THIS IS NOT THE RIG
     HT TREE": PRINT
10090  PRINT "FIRST PLAYER TO PIC
     K 15 APPLES IS THE": PRINT "
     WINNER": PRINT : PRINT "HIT
     ANY KEY AND GOOD PICKING";
10100  GET A$
10110  PRINT "THE APPLE. THE FIRS
     T PLAYER TO SCORE": PRINT "1
     5 POINTS IS THE WINNER.": PRINT

10140  RETURN
```

Initialization; load variables; poke tone routine; poke arrow shape.

```
20000  HOME : INVERSE : VTAB 7: HTAB
     13: PRINT "APPLE  CAPTURE": PRINT
     : HTAB 19: PRINT "BY": PRINT
     : HTAB 15: PRINT "BILL  RYAN
     ": PRINT
20050  FOR Z = 1 TO 3000: NEXT Z
20060  FOR Z = 0 TO 9: FOR Q = 1 TO
     7: READ S(Z,Q): NEXT Q,Z
20070  FOR Z = 24576 TO 24613: READ
     Q: POKE Z,Q: NEXT Z
20080  POKE 232,0: POKE 233,96
20082  POKE 770,173: POKE 771,48:
      POKE 772,192: POKE 773,136:
      POKE 774,208: POKE 775,5: POKE
     776,206: POKE 777,1: POKE 77
     8,3: POKE 779,240: POKE 780,
     9: POKE 781,202
20084  POKE 782,208: POKE 783,245
     : POKE 784,174: POKE 785,0: POKE
     786,3: POKE 787,76: POKE 788
     ,2: POKE 789,3: POKE 790,96:
      POKE 791,0: POKE 792,0
20089 SC(1) = 0:SC(2) = 0
20090  NORMAL : RETURN
```

Data for segments.

```
21000  DATA  1,2,3,5,6,7,0
21001  DATA  3,6,0,0,0,0,0
21002  DATA  1,3,4,5,7,0,0
21003  DATA  1,3,4,6,7,0,0
21004  DATA  2,4,3,6,0,0,0
21005  DATA  1,2,4,6,7,0,0
21006  DATA  1,2,4,5,6,7,0
21007  DATA  1,3,6,0,0,0,0
21008  DATA  1,2,3,4,5,6,7
21009  DATA  1,2,3,4,6,7,0
```

Data for arrow shape.

```
21040  DATA  1,0,4,0,63,63,63,46,
     45,45,45,45,53,62,62,62,62,3
     7,37,37,37,37,37,39,39,39,39
     ,39,47,46,46,46,46,46,63,63,
     63,0
```

SoftSide Sentinel

NEW YORK, FRIDAY JANUARY 1, 1982

CITY EDITION

Metropolitan area weather: Cloudy today, tonight. Partly sunny tomorrow. Temperature range: today 44-55; yesterday 48-54. Details, page B8.

35 CENTS

"All the Software that's fit to print"

VOL. CXXXI....No. 45,135

INVESTIGATION CONTINUES IN HOTEL BOMB THREAT

BY GARY J. DOMINICK

***Piazza Hotel* is a logical deduction game program for an Atari with 16K RAM.**

As the house detective of the Piazza Hotel, you have just been informed that there is a bomb in one of the rooms. Your job is to find it — FAST.

You begin your search by choosing a floor and a room number. If you are not correct, you will be given clues to help you narrow your search. You never know what the next guess will lead to. Sometimes it's the correct room. Sometimes it's. . . KABOOM!

Variables

B, D, F, G, J: Temporary variables used in explosion sequence.
BOOM: Random number to determine when the bomb will explode.
COFL: Floor number chosen at random by the computer.
CORM: Room number chosen at random by the computer.
FLC: Number used to determine floor clue.
H: Horizontal position of cursor to draw bomb.
PLFL: Floor number guessed by the player.
PLRM: Room number guessed by the player.
RATE$: String variable used to print rating.
RMC: Number used to determine room clue.
V: Vertical position of cursor to draw bomb.
W, WW, WWW: Subroutines used for delays.
U, X, Y, Z, ZZ: Temporary variables used in FOR-NEXT loops.

Site of Piazza Hotel where a mysterious b located in one of the various rooms.

House Detec Seeks Clu

BY COLLEEN NE

NEW YORK — Piazza Hote detectives continued their today for a bomb planted rooms. "Our concern at find the bomb before it b we will seek suspects, tive, who asked for anc investigation is comple using a cor he add

Senate Endorses m for School

```
$$$$$$$$$$$$$$$$$$$$$$$$$
$      Atari BASIC      $
$    'PIAZZA HOTEL'     $
$ AUTHOR: Gary Dominick $
$   (c) 1982 SoftSide   $
$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialize subroutine line numbers.

```
10 TI=1000:HO=4000:CS=5000:PG=6000:CA=
7000:CL=8000:CO=9000:EX=10000
15 RIGHT=8100:LEFT=8200:UP=8300:DOWN=8
400:W=11200:WW=11200:WWW=11000:DIM RAT
E$(30)
```

Opens keyboard.

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

The main program which consists of five subroutines.

```
100 GOSUB TI
400 GOSUB HO
500 GOSUB CS
600 GOSUB PG
700 GOSUB CA
990 GOTO 600
```

Routine to create the flashing marquee title graphics.

```
1000 REM ** TITLE GRAPHIC **
1010 GRAPHICS 2+16
1020 COLOR 1:POSITION 5,5:? #6;"PIAZZA
 HOTEL"
1200 FOR U=1 TO 5:FOR X=2 TO 18
1210 COLOR 2:SETCOLOR 1,4,14:POSITION
X,3:? #6;CHR$(270):POSITION 20-X,7:? #
6;CHR$(270)
1220 COLOR 1:POSITION X,3:? #6;CHR$(14
2):POSITION 20-X,7:? #6;CHR$(142)
1230 FOR Z=1 TO 8:NEXT Z:NEXT X
1310 FOR ZZ=4 TO 6:COLOR 2:SETCOLOR 1,
0,14:POSITION 2,10-ZZ:? #6;CHR$(270):P
OSITION 18,ZZ:? #6;CHR$(270)
1320 COLOR 1:POSITION 2,10-ZZ:? #6;CHR
$(142):POSITION 18,ZZ:? #6;CHR$(142):F
OR Z=1 TO 16:NEXT Z:NEXT ZZ
1330 NEXT U
1999 RETURN
```

Routine to draw the hotel in different colors each game.

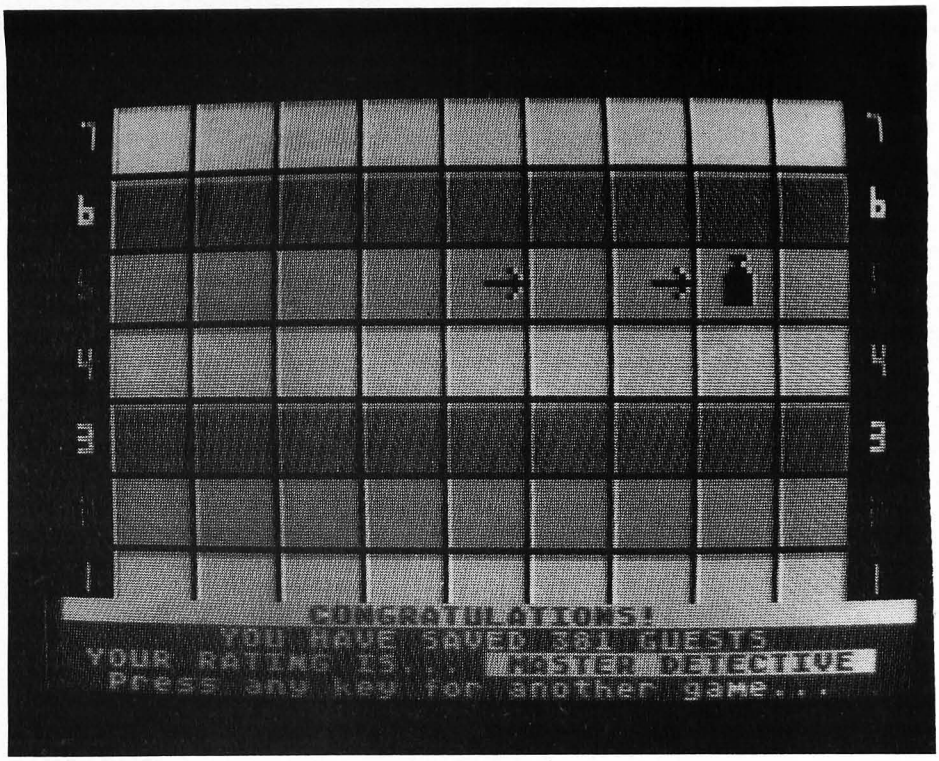
```
4000 REM ** DRAW HOTEL **
```

Draws bands of different shades of one color to represent each floor of the hotel.

```
4010 GRAPHICS 7:COLOR 1:D=INT(RND(0)*1
5)+1:IF D=3 OR D=4 OR D=5 THEN GOTO 40
10
4020 FOR X=0 TO 77 STEP 12
4030 C=C+1:IF C=4 THEN C=1
4040 SETCOLOR 0,D,6:SETCOLOR 1,D,10:SE
TCOLOR 2,D,2
4100 PLOT 145,10+X:DRAWTO 145,0+X:DRAW
TO 13,0+X:POSITION 13,10+X
4110 POKE 765,C
4120 XIO 18,#6,0,0,"S:"
4130 NEXT X
```

Draws the numbers 1 to 7 in graphics 7 on the left and right side of the hotel.

```
4300 REM ** DRAW FLOOR NUMBERS **
4305 FOR X=2 TO 145 STEP 143
4310 COLOR 1:PLOT 6+X,74:DRAWTO 6+X,76
4320 PLOT 5+X,39:DRAWTO 5+X,41:DRAWTO
7+X,41:PLOT 7+X,39:DRAWTO 7+X,43
4330 PLOT 5+X,2:DRAWTO 7+X,2:DRAWTO 7+
X,6
4340 COLOR 3:PLOT 5+X,63:DRAWTO 7+X,63
:DRAWTO 7+X,65:DRAWTO 5+X,65:DRAWTO 5+
X,67:DRAWTO 7+X,67
4350 PLOT 7+X,26:DRAWTO 5+X,26:DRAWTO
5+X,28:DRAWTO 7+X,28:DRAWTO 7+X,30:DRA
WTO 5+X,30
4360 COLOR 2:PLOT 5+X,51:DRAWTO 7+X,51
:DRAWTO 7+X,55:DRAWTO 5+X,55:PLOT 5+X,
53:DRAWTO 7+X,53
4370 PLOT 5+X,14:DRAWTO 5+X,18:DRAWTO
7+X,18:DRAWTO 7+X,16:DRAWTO 5+X,16
4372 NEXT X
```



Divides each floor into nine rooms.

```
4375 REM ** DIVIDE ROOMS **
4380 FOR Y=15 TO 143 STEP 15
4390 COLOR 0:PLOT 12+Y,0:DRAWTO 12+Y,7
9
4395 NEXT Y
```

Prints room numbers and name of hotel in the text window under the hotel.

```
4400 REM ** ROOM NUMBERS **
4410 POKE 752,1
4420 POKE 656,0:POKE 657,3:? "  1  2
 3  4  5  6 7  8 9 "
4430 POKE 656,1:POKE 657,1:? " PIAZZA
HOTEL PIAZZA HOTEL PIAZZA HOTEL"
4999 RETURN
5000 REM ** COMPUTER SELECTION **
```

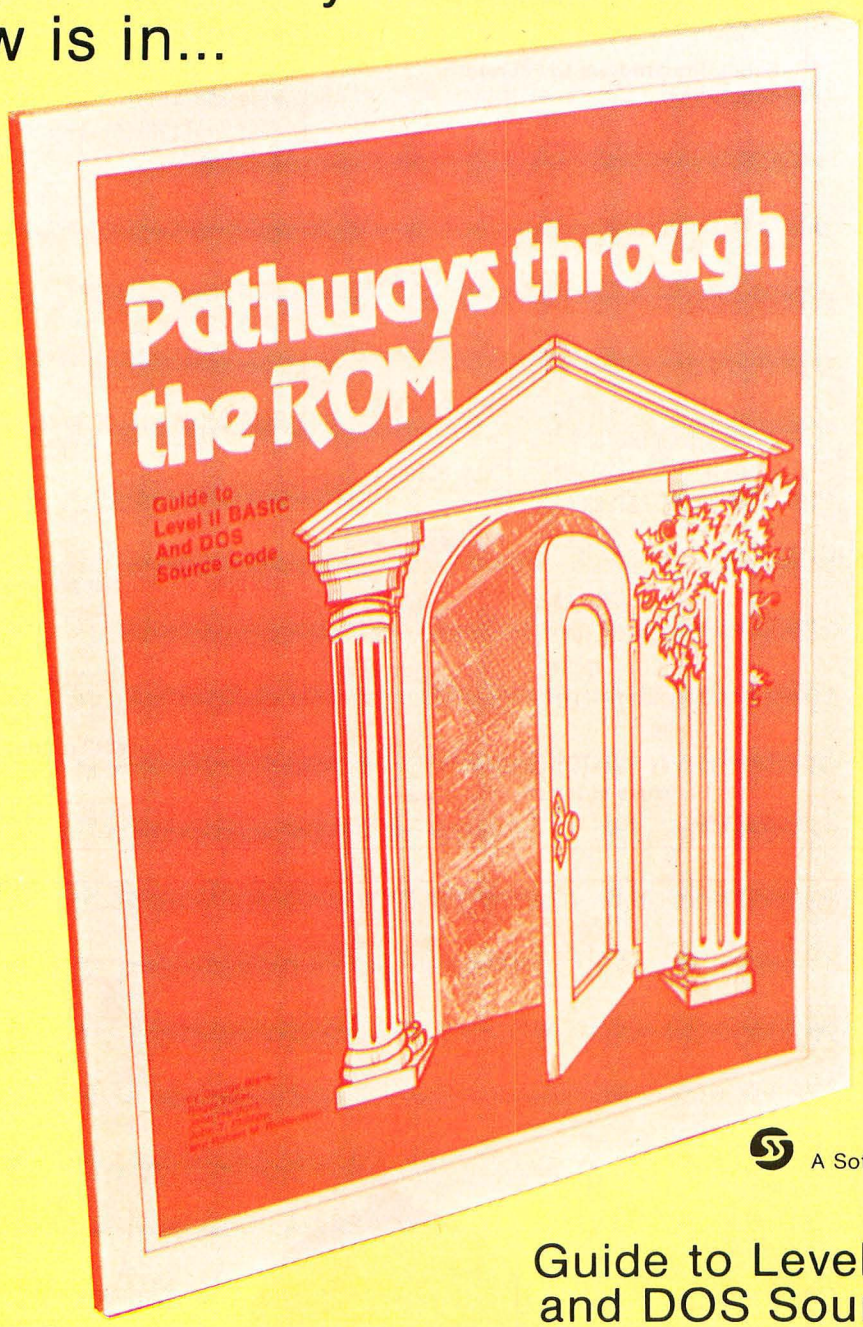
Computer chooses floor and room numbers where bomb is hidden.

```
5100 COFL=INT(RND(0)*7)+1:CORM=INT(RND
(0)*9)+1
```

Computer chooses how many guesses the player will have.

```
5200 BOOM=INT(RND(0)*5)+2
5300 EXPL=0:GUESS=0
5400 RETURN
6000 REM ** PLAYER GUESS **
6010 GOSUB WW:POKE 656,3:POKE 657,2:?
"                                    ";
```

Unlock the hidden power of your computer for fast and easy programming! Use ROM routines in your BASIC and Assembly Language programs! All you need to know is in...



INCLUDES:

SUPERMAP
From Fuller Software ($18.95)

TRS-80 DISASSEMBLED HANDBOOK
by Robert Richardson ($10.00)

HEX MEM
by John Phillipp
Monitor written in BASIC

Z-80 DISASSEMBLER
by George Blank

ALL ONLY $19.95
plus $1 shipping

A SoftSide Publication

Guide to Level II BASIC and DOS Source Code

Description of the contents of the Level II BASIC ROM by memory locations, by function, and in lesson format. Includes several BASIC and Assembly Language programs in listing format to examine and use ROM routines.

MasterCard VISA

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

```
6100 POKE 656,3:POKE 657,3:SOUND 0,1,6
,12:GOSUB W:SOUND 0,0,0,0:? "WHICH FLO
OR ?";
```

Gets floor number of player's guess. If not between 1 and 7, erases guess and asks again.

```
6110 GET #2,FL:IF FL<49 OR FL>55 THEN
SOUND 0,20,6,12:GOSUB W:SOUND 0,0,0,0:
GOTO 6010
```

Converts ATASCII number generated by GET command to decimal number.

```
6115 PLFL=FL-48
```

Repeats procedure for room guess.

```
6117 POKE 656,3:POKE 657,19:? "
          ";
6120 SOUND 0,3,6,12:GOSUB W:SOUND 0,0,
0,0:POKE 656,3:POKE 657,17:? PLFL;"
WHICH ROOM ?";
6130 GET #2,RM:IF RM<49 OR RM>57 THEN
SOUND 0,20,6,12:GOSUB W:SOUND 0,0,0,0:
GOTO 6117
6140 PLRM=RM-48:SOUND 0,2,6,12:GOSUB W
:SOUND 0,0,0,0:POKE 656,3:POKE 657,34:
? PLRM;:GOSUB WW
6999 RETURN
```

Routine used after each guess to determine what clues to give.

```
7000 REM ** CLUE CALCULATION **
```

Decides if the bomb should blow up or not.

```
7100 EXPL=EXPL+1:GUESS=GUESS+1:IF EXPL
=BOOM THEN GOSUB EX
```

Based on the subtraction of the player's guess from the computer selection, the program will branch to the subroutine to print the clues.

```
7150 FLC=COFL-PLFL:RMC=CORM-PLRM
```

Sends the program to "correct" subroutine.

```
7200 IF FLC=0 AND RMC=0 THEN GOSUB CO:
RETURN
7250 IF FLC=0 AND RMC>0 THEN GOSUB RIG
HT:RETURN
7300 IF FLC=0 AND RMC<0 THEN GOSUB LEF
T:RETURN
7350 IF FLC>0 AND RMC=0 THEN GOSUB UP:
RETURN
7400 IF FLC>0 AND RMC>0 THEN GOSUB RIG
HT:GOSUB UP:RETURN
7450 IF FLC>0 AND RMC<0 THEN GOSUB LEF
T:GOSUB UP:RETURN
7500 IF FLC<0 AND RMC=0 THEN GOSUB DOW
N:RETURN
7550 IF FLC<0 AND RMC>0 THEN GOSUB RIG
HT:GOSUB DOWN:RETURN
7600 IF FLC<0 AND RMC<0 THEN GOSUB LEF
T:GOSUB DOWN:RETURN
7999 RETURN
8000 REM ** PRINTED CLUES **
```

Draws the clues in the room that was guessed.

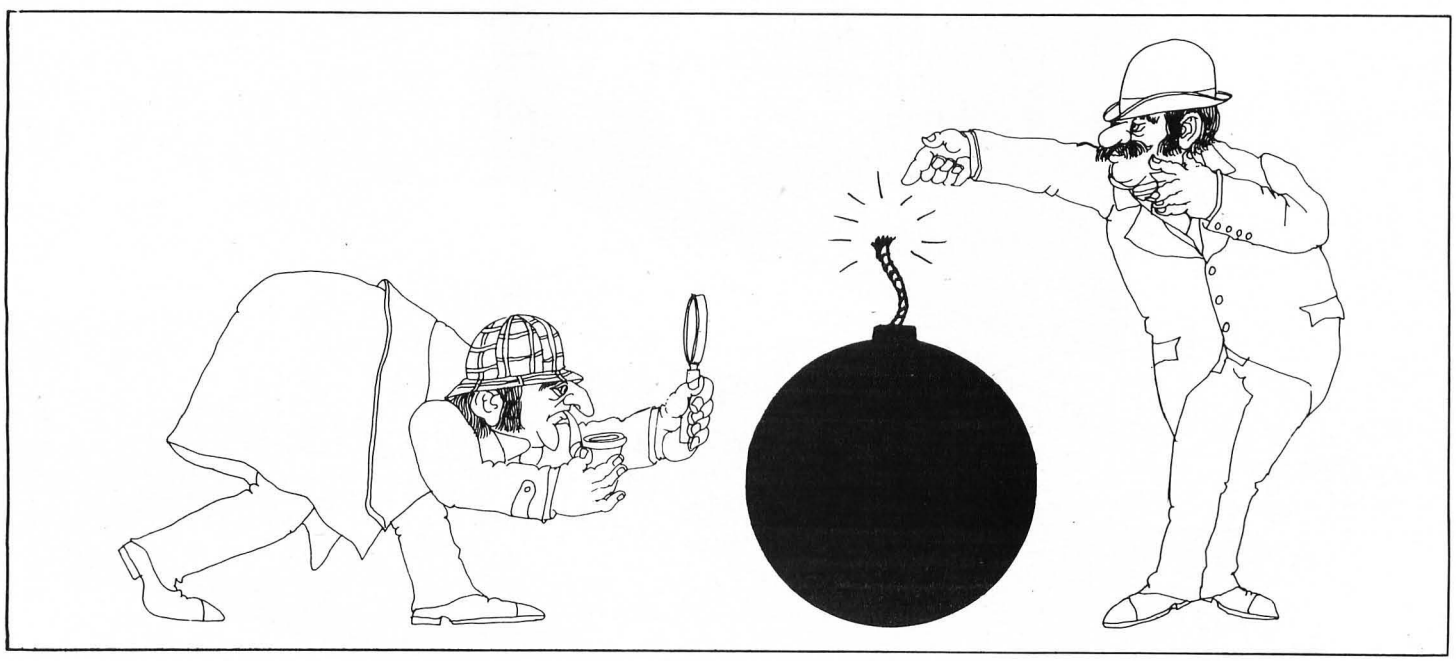
```
8100 REM ** RIGHT **
8105 H=(PLRM*15)+5:V=89-(12*PLFL):GOSU
B 8500
8110 COLOR 0:PLOT H,V:DRAWTO H+6,V
8120 PLOT H+4,V+2:DRAWTO H+6,V:DRAWTO
H+4,V-2
8130 H=0:V=0:RETURN
8200 REM ** LEFT **
8205 H=(PLRM*15)+4:V=89-(12*PLFL):GOSU
B 8500
8210 COLOR 0:PLOT H,V:DRAWTO H-6,V
8220 PLOT H-4,V-2:DRAWTO H-6,V:DRAWTO
H-4,V+2
8230 H=0:V=0:RETURN
8300 REM **  UP  **
8305 H=(PLRM*15)+4:V=89-(12*PLFL):GOSU
B 8500
8310 COLOR 0:PLOT H,V:DRAWTO H,V-5
8320 PLOT H-2,V-3:DRAWTO H,V-5:DRAWTO
H+2,V-3
8330 H=0:V=0:RETURN
8400 REM ** DOWN **
8405 H=(PLRM*15)+4:V=89-(12*PLFL):GOSU
B 8500
8410 COLOR 0:PLOT H,V:DRAWTO H,V+5
8420 PLOT H-2,V+3:DRAWTO H,V+5:DRAWTO
H+2,V+3
8430 H=0:V=0:RETURN
```

Creates sound after printing each clue.

```
8500 FOR M=1 TO 7
8510 SOUND 0,40+(M*10),10,12
8520 NEXT M:SOUND 0,0,0,0:RETURN
```

Flashing "correct" routine.

```
9000 REM ** CORRECT ROUTINE **
```



Draws bomb in room.

```
9010 H=(CORM*15)+5:V=90-(12*COFL)
9100 FOR Z=-2 TO 2:COLOR 0:PLOT H-2,V+
Z:DRAWTO H+2,V+Z:NEXT Z
9110 PLOT H-1,V-3:DRAWTO H+1,V-3:PLOT
H-1,V-5:DRAWTO H+1,V-5:PLOT H,V-5:DRAW
TO H,V-3
9200 FOR Z=1 TO 8
9220 SETCOLOR 4,0,14:SOUND 0,50,10,12:
GOSUB WW:SETCOLOR 4,0,0:SOUND 0,150,10
,12:GOSUB WW
9230 NEXT Z:SOUND 0,0,0,0
9300 FOR Z=0 TO 3:POKE 656,Z:POKE 657,
1:? "
     ";:NEXT Z
```

End of game results.

```
9310 POKE 752,1:POKE 656,0:POKE 657,1:
? "          CONGRATULATIONS!
   ";
9320 GUESTS=INT(RND(0)*100)+(600-GUESS
*100)
9330 POKE 656,1:POKE 657,8:? "YOU HAVE
 SAVED ";GUESTS;" GUESTS";
```

Determines rating based on the number of guesses it took to find the bomb.

```
9340 IF GUESS=1 OR GUESS=2 THEN RATE$=
" SUPER SLEUTH "
9344 IF GUESS=3 THEN RATE$=" MASTER DE
TECTIVE "
9346 IF GUESS=4 THEN RATE$="DETECTIVE
CLASS-A"
9348 IF GUESS=5 THEN RATE$="DETECTIVE-
CLASS B"
9350 IF GUESS>5 THEN RATE$=" GUM - SHO
E "
9360 POKE 656,2:POKE 657,2:? "YOUR RAT
ING IS... ";:? RATE$;
9400 POKE 656,3:POKE 657,3:? "Press an
y key for another game...";
9410 GET #2,AGAIN:GOTO 400
```

Explosion routine.

```
10000 REM ** EXPLOSION **
```

Draws bomb in room where it was placed by the computer.

```
10090 H=(CORM*15)+5:V=90-(12*COFL)
10100 FOR Z=-2 TO 2:COLOR 0:PLOT H-2,V
+Z:DRAWTO H+2,V+Z:NEXT Z
10110 PLOT H-1,V-3:DRAWTO H+1,V-3:PLOT
 H-1,V-5:DRAWTO H+1,V-5:PLOT H,V-4:DRA
WTO H,V-3
```

Creates visual and sound effects for explosion.

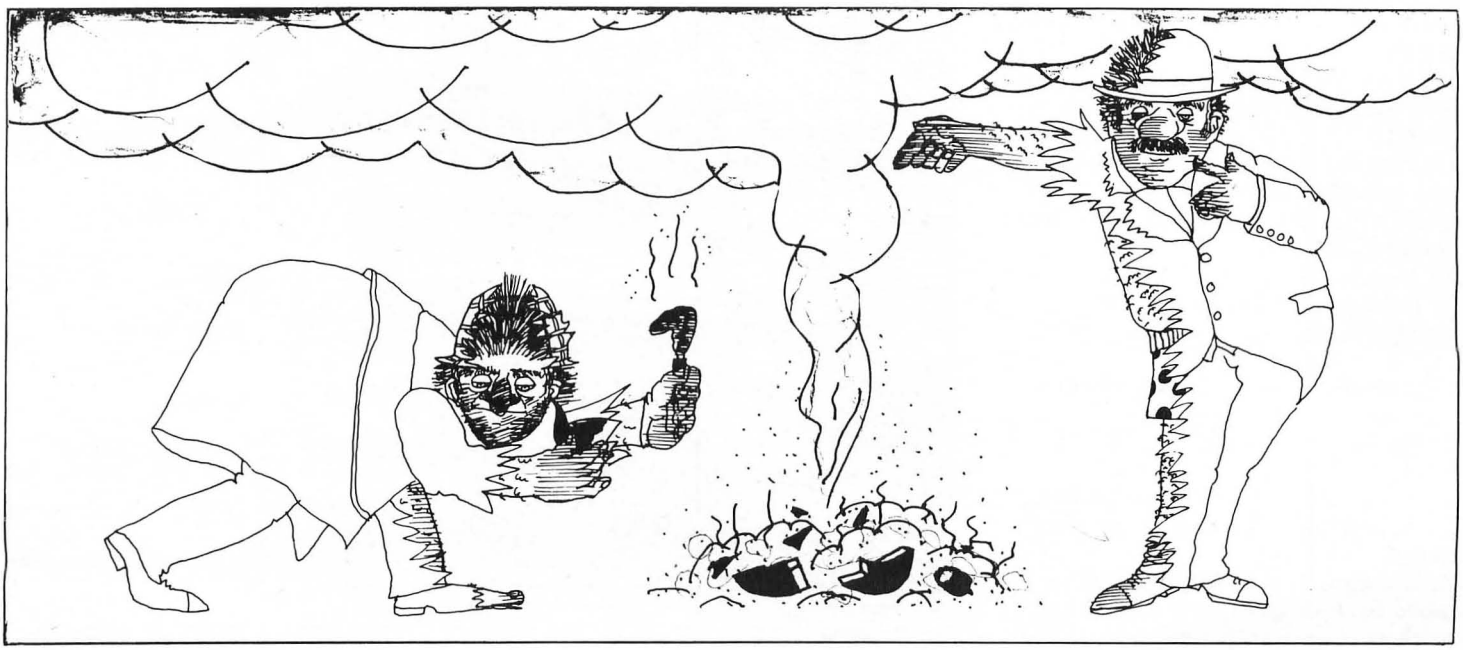
```
10200 GOSUB W:SOUND 2,75,8,15:V1=15:V2
=15:V3=15:FLASH=2
10210 SOUND 0,20,8,V1:SOUND 1,40,8,V2:
SOUND 2,70,8,V3
10220 COLOR 0:SETCOLOR 4,4,FLASH:B=INT
(RND(0)*H)+1:G=INT(RND(0)*V)+1:J=INT(R
ND(0)*H)+1:F=INT(RND(0)*V)+1
10225 FLASH=FLASH+12:IF FLASH>14 THEN
FLASH=2
```

Makes sure that rays don't go out of range.

```
10226 IF H+B>158 THEN B=0
10227 IF V-G<0 THEN G=0
10228 IF V-F<0 THEN F=0
10229 IF H-J<0 THEN C=0
10230 PLOT H,V+2:DRAWTO H+B,V+2-G:PLOT
 H,V+2:DRAWTO H-J,V+2-F
10250 V1=V1*0.91:V2=V2*0.95:V3=V3*0.97
:IF V3>3.2 THEN 10210
10270 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUN
D 2,0,0,0
10300 POKE 752,1
10310 IF COFL=PLFL AND CORM=PLRM THEN
GOSUB 12000:GOTO 10330
10320 POKE 656,0:POKE 657,0:? "
  TOO BAD! ---- TOO LATE!          "
10330 POKE 656,1:POKE 657,0:? "  YOU A
RE NOW A DETECTIVE IN THAT BIG  "
10340 POKE 656,2:POKE 657,0:? "   HOTE
L IN THE SKY.TO RETURN TO EARTH  ";
10350 POKE 656,3:POKE 657,0:? "    FOR
 ANOTHER GAME PRESS ANY KEY...  ";
10400 GET #2,Q:GOTO 400
```

Time-delay subroutines.

```
11000 FOR M=1 TO 100:NEXT M
11100 FOR M=1 TO 50:NEXT M
11200 FOR M=1 TO 50:NEXT M:RETURN
12000 POKE 656,0:POKE 657,0:? "   RIGH
T ROOM -JUST TOO LATE- TOO BAD!  ";RET
URN
```



Number Race

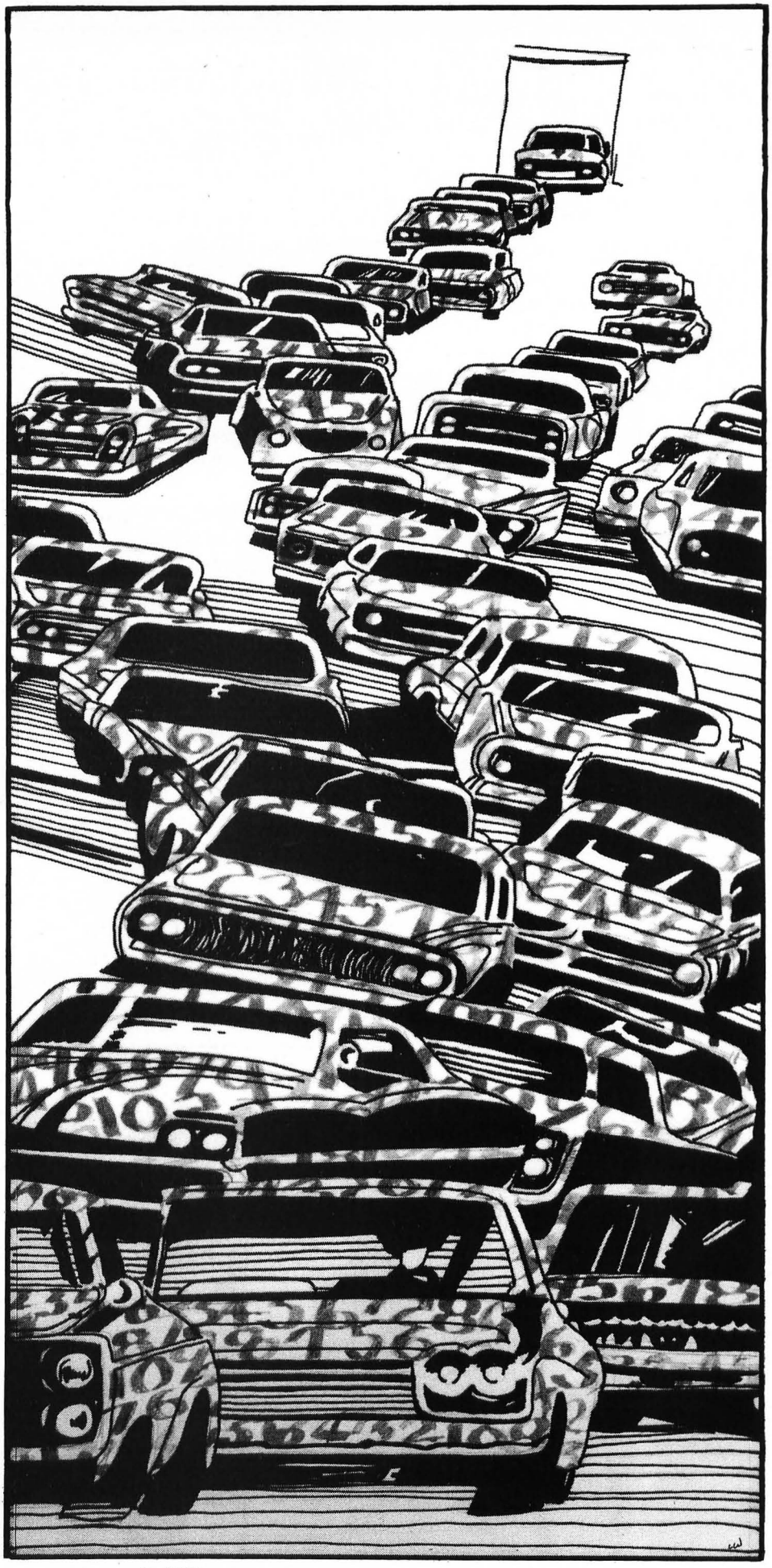
by Arthur N. Schreibman

***Number Race* is a game program for an Atari with 8K or more RAM and one joystick.**

A random string of numbers moves from right to left across the lower part of the screen between two large blue asterisks. In the center of the screen is a single digit that you can increase by moving the joystick in any direction. When your number matches one of the digits in the string below, hit the fire button. The number will vanish from the string, the other numbers will move backwards to eliminate the space of the vanished number, and you will be awarded points. Continue until you have cleared the string, or until one of the numbers reaches the left asterisk. You will be razzed or trumpeted, depending on how well you did. After a couple of seconds, round two begins. This time the numbers march out a little faster, but the point value for each match increases. The fifth and final round is difficult enough to test the best. A perfect score is 1000 points. High score for each session is held on the screen.

Variables

GAME: Flag for mid-game (=1), end-game (=0).
H: ATASCII value of number.
HTOT: High game score.
I, J: Loop counters.
K: Column used to get numbers to move across screen and to check for a match.
M: Value of matching number.
N$: Number string to match each round.
Q: Flag for joystick movement (=1), and return to center (=0).
QUIT: Flag for lost round (=1), won round (=0).
R: Number of round.
S: Random number to generate number string.
SC: Round score.
ST$: String version of each random number.
T: Timer loop counter.
TOT: Cumulative game score.
V, W: Position of matching number.
X, Y: Positions for moving number string.
Z: Column used to get numbers to move backwards after a match, and to clear row after game.



```
$$$$$$$$$$$$$$$$$$$$$$$$$
$      Atari BASIC      $
$     'NUMBER RACE'     $
$ AUTHOR: A. Schreibman $
$   (c) 1982 SoftSide   $
$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization.

```
50 GRAPHICS 2
55 HTOT=0:TOT=0
60 DIM N$(16),ST$(1):POKE 752,1
65 R=1:GAME=0:QUIT=0
```

Prints playfield in graphics area and scoreboard in text window. Line 70 has alternate letters of the string in inverse video. The GAME = 1 flag in line 73 changes the round score back to 0 after each round. When the game is over the GAME = 1 (line 1260) and the score of the final round will remain on the screen. Line 97 brings scores back to 0 at the start of a new game. The last character in the string in line 97 is a control character vertical arrow.

```
70 POSITION 4,0:? #6;"number race"
73 IF GAME=1 THEN SC=0
75 POSITION 1,2:? #6;"ROUND ";R;"  "
77 POSITION 6,7:? #6;"*":POSITION 13,7
:? #6;"*"
80 POKE 656,1:POKE 657,1:? "ROUND SCOR
E: ";SC;"  "
85 POKE 656,1:POKE 657,28:? "HIGH: ";H
TOT
90 POKE 656,2:POKE 657,1:? "TOTAL SCOR
E: ";TOT;"  "
92 IF GAME=1 THEN GOTO 108
96 POKE 656,3:POKE 657,8:? "PRESS 'FIR
E' TO START";CHR$(28)
97 IF STRIG(0)=0 THEN SC=0:TOT=0:POKE
656,1:POKE 657,14:? SC;"  ";:POKE 656
,2:POKE 657,14:? TOT;"  ":GOTO 108
98 GOTO 97
```

Fills the number string with ten digits. Line 110 converts the number S into a string, ST$.

```
108 FOR I=1 TO 10
110 S=INT(RND(0)*10+1):ST$=STR$(S)
120 N$(I,I)=ST$
130 NEXT I
140 FOR I=11 TO 16
150 N$(I,I)=" "
160 NEXT I
```

Moves numbers across screen and changes your matching number. Line 245 allows the stick to increment the matching number, M. Q is set to 1 indicating the change. Line 255 sets Q back to 0 if the joystick goes back to the center position. This allows the program to pass through line 243 and increment M again. Without this procedure, holding the joystick off-center would continually change the value of M instead of increasing M one digit for each joystick move. The string row is always 7, and Y is set to that constant. The first digit starts out in column 12 and moves left to column 7. Line 230 prints the first digit in position 12,7. Line 240 determines the speed of the game, getting faster as the rounds, R, progress. Increasing the 70 in that line will slow down the game. Line 275 finds the value, H, in each position and line 280 prints those characters one position to the left. Line 285 checks to see if a digit reached the last position on the left, and if it did, the lost round flag, QUIT, is set to 1. If the round ends, line 290 sends the program to line 1200.

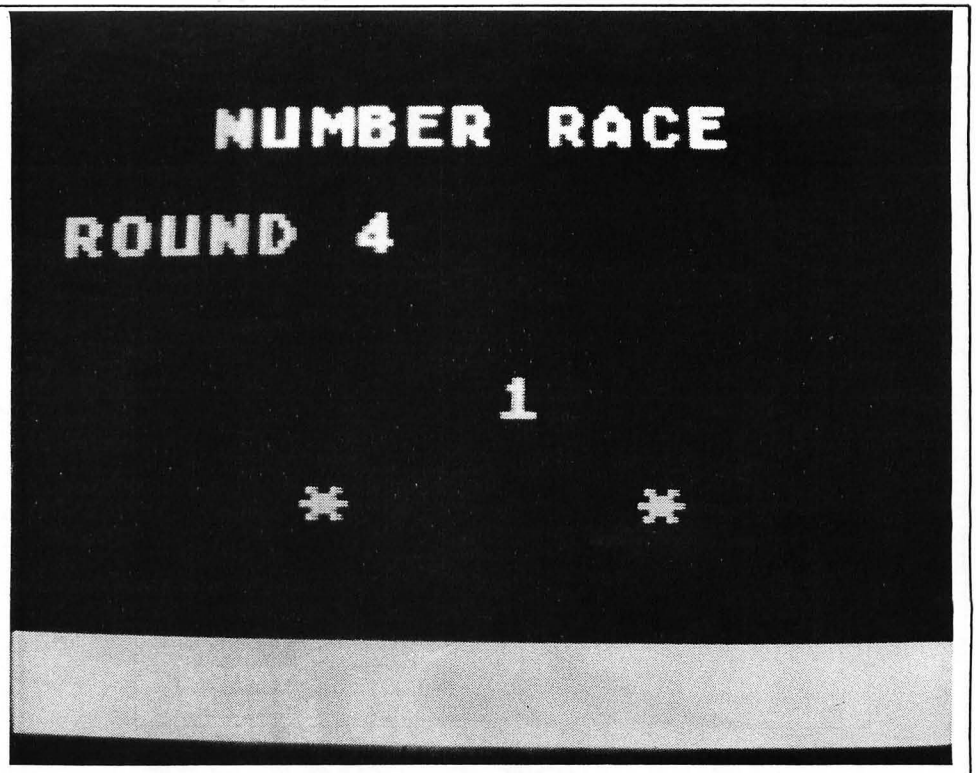
```
200 X=12:Y=7:M=0:V=10:W=5
205 POSITION V,W:? #6;M
207 SOUND 0,40,10,6:FOR T=0 TO 20:NEXT
 T:SOUND 0,0,0,0
210 FOR J=1 TO 16
230 POSITION X,Y:? #6;N$(J,J)
235 SOUND 0,100,10,6:FOR T=0 TO 20:NEX
T T:SOUND 0,0,0,0
240 FOR T=0 TO 70-(15*(R-1))
243 IF Q=1 THEN 246
245 IF STICK(0)<>15 THEN M=M+1:Q=1:POK
E 77,0:SOUND 0,40,10,6:FOR WT=0 TO 20:
NEXT WT:SOUND 0,0,0,0
246 IF M=10 THEN M=0
247 POSITION V,W:? #6;M
250 IF STRIG(0)=0 THEN 1000
255 IF STICK(0)=15 THEN Q=0
260 NEXT T
270 FOR K=8 TO 12
275 LOCATE K,Y,H
280 POSITION K-1,Y:? #6;CHR$(H)
285 LOCATE 7,Y,H:IF H<>32 THEN QUIT=1
290 NEXT K:IF QUIT=1 THEN 1200
300 NEXT J
```

Line 1010 checks the ATASCII value of each digit in the string if the fire button is pressed (from line 250) and compares it to M in line 1020. If there is a match, lines 1030 to 1055 move all the characters to the left of the match one position to the right.

```
1000 FOR K=8 TO 12
1010 LOCATE K,Y,H
1020 IF H=(M+48) THEN 1027
1025 NEXT K
1026 GOTO 270
1027 SOUND 0,180,10,10:FOR T=0 TO 15:N
EXT T
1028 SOUND 0,120,10,10:FOR T=0 TO 15:N
EXT T:SOUND 0,0,0,0
1030 FOR Z=K-1 TO 7 STEP -1
1040 LOCATE Z,Y,H
1050 POSITION Z+1,Y:? #6;CHR$(H)
1052 POSITION Z,Y:? #6;" "
1055 NEXT Z
```

Increments the score.

```
1080 SC=SC+10+5*(R-1)
1090 POKE 656,1:POKE 657,14:? SC
```



```
1095 IF SC=10*(10+5*(R-1)) THEN 1200
1100 GOTO 270
```

Cleans up things at the end of a round. Line 1200 increments the round. Line 1210 adds the round score to the game score and it is printed in line 1215. Lines 1225 to 1240 blank the area between the asterisks after a round. Line 1260 sets GAME = 1 to signify we are in the middle of the game and sends the program back to start the next round.

```
1200 R=R+1
1210 TOT=TOT+SC
1215 POKE 656,2:POKE 657,1:? "TOTAL SC
ORE: ";TOT;"   "
1217 IF QUIT=1 THEN SOUND 0,120,12,15:
FOR T=0 TO 200:NEXT T:SOUND 0,0,0,0:GO
TO 1224
1220 SOUND 0,130,10,8:FOR T=0 TO 65:NE
XT T
1221 SOUND 0,122,10,8:FOR T=0 TO 65:NE
XT T:SOUND 0,155,10,12:FOR T=0 TO 100:
NEXT T
1222 SOUND 0,100,10,12:FOR T=0 TO 180:
NEXT T:SOUND 0,0,0,0
1224 QUIT=0
1225 FOR Z=7 TO 12
1230 POSITION Z,Y:? #6;" "
1240 NEXT Z:IF R=6 THEN 1300
1250 FOR T=0 TO 450:NEXT T
1260 GAME=1:GOTO 73
```

End game. Line 1310 keeps track of the high score for the session and 1350 sends the program back to initialize for the next game.

```
1300 POSITION 1,2:? #6;"GAME OVER"
1310 IF TOT>HTOT THEN HTOT=TOT
1350 FOR D=1 TO 1000:NEXT D:GOTO 65
```

Atari One Liners

```
1 POKE 752,1:PRINT"  ";:R=28+RND(0)*4:
PRINT CHR$(R);CHR$(R+1-(R>30));" V";:F
OR I=1 TO RND(0)*12:SOUND 0,I*2,10,4:N
EXT I:GOTO 1
```

Nick Buehler
Andover, MA

```
1 GRAPHICS 56:C=RND(0)*60:POKE 710,C*2
:COLOR 1:FOR Z=0 TO 319 STEP 8:PLOT 15
9,95:DRAWTO X,SIN(X/C)*C+95:NEXT X:GOT
O 1
```

Lynn Wallace
Rapid City, SD

CALENDAR

January 6-8

Fifteenth Annual Hawaii International Conference on Systems Sciences (HICSS-15)
Honolulu, HI

Sponsored by the University of Hawaii and the University of Southwestern Louisiana (in cooperation with the Association for Computing Machinery), this conference is directed at medical information-processing researchers and practitioners. Topics to be covered include diagnosis by the computer, computer-based medical instrumentation, and computers and the handicapped. The use of computers in both individual and group practices, as well as medical labs and hospitals will also be covered in this three-day session.
Contact: Dr. Bruce D. Shriver, c/o HICSS-15 Medical Information Processing, University of Southwestern Louisiana, P.O. Box 44330, Lafayette, LA 70504

January 7-10

1982 Winter Consumer Electronics Show (CES)
Las Vegas Convention Center, Hilton Hotel and Jockey Club, Las Vegas, NV

Seminars and workshops will be the highlights of this conference. Audio and video equipment, computers, and other consumer items will be exhibited at this event with over 800 vendors represented.
Contact: Consumer Electronics Shows, Two Illinois Center, Suite 1607, 233 N. Michigan, Chicago, IL 60601, (312)861-1040

January 12-15

Communication Networks Conference and Exhibition
Georgia World Congress Center, Atlanta, GA

This conference is designed for people in the telecommunications industry. Sessions will include panel discussions, and tutorials on voice, data, and electronic mail communications.
Contact: Communication Networks, 375 Cochituate Road, P.O. Box 880, Framingham, MA 01701, (617)879-0700

January 14

Invitational Computer Conference
S. Coast Plaza Hotel, Orange County, CA

This conference will include seminars and displays of computer and peripheral equipment. The focus will be aimed at the needs of the quantity buyer.
Contact: B. J. Johnson and Associates, 2503 Eastbluff Drive, Suite 203, Newport Beach, CA 92660

January 15-16

Microcomputers in Education: Uses for the 80s
Arizona State University, Tempe, AZ

This is the tenth annual conference on math and science. It will stress the microcomputer as an instructional tool which can also be used for research and as an information manager. Workshops, demonstrations, panel discussions, and problem-solving groups will be offered.
Contact: Nancy Watson, 203 Payne Hall, Arizona State University, Tempe, AZ 85287

January 19-22

Peripheral Array Processors for Signal Processing and Simulation
Sheraton National Hotel, Washington, DC

Details for this seminar were not available at press time. The fee for this course is $795. Please write for details.
Contact: Continuing Education Institute, 10889 Wilshire Boulevard, Suite 1030, Los Angeles, CA 90024, (213)824-9545

January 20-22

Texas Computer Show
Dallas Convention Center, Dallas, TX

Word- and data-processing equipment will be featured in demonstrations. Panel discussions, seminars, and conferences will be an added enhancement to the show.
Contact: Texas Computer Show, P.O. Box 214035, Dallas, TX 75221, (214)761-9108

January 28-30

Conference on Modeling and Simulation on Microcomputers
Bahia Hotel, San Diego, CA

This conference is sponsored by the Society for Computer Simulation (SCS). Discreet and continuous simulation on microcomputers will include papers, panel discussions, and tutorials.
Contact: SCS, P.O. Box 2228, La Jolla, CA 92038, (714)459-3888

Short Courses in Advanced Technology

This self-training program was developed for and by engineers. It is a direct extension of the material used in the ICS classroom microprocessor courses. A total of 12 different courses will be offered in advanced technology over the next few months. Special in-plant presentations can be scheduled at your place of business. Other courses in microprocessor self-training and video courses will be offered. Please write for free brochure and complete schedule of classes and prices. The dates and course titles for this month are as follows:

January 19-22

Digital Filters and Spectral Analysis
San Francisco, CA
Hands-On Pascal Workshop
Washington, DC
Computerized Robots
Washington, DC

January 26-29

Microprocessor Software, Hardware and Interfacing
Los Angeles, CA
Computer Network Design and Protocols
San Francisco, CA
Digital Filters and Spectral Analysis
Washington, DC
Hands-on Microprocessor Troubleshooting
Los Angeles, CA
Contact: Integrated Computer Systems, 3304 Pico Boulevard, P.O. Box 5339, Santa Monica, CA 90405, (800)421-8166, in California (800)352-8251

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.

BOUND TO PLEASE: SoftSide's Deluxe Edition

Twelve issues of SoftSide — October 1980 to September 1981

SoftSide is pleased to announce this limited edition: twelve issues sewn into blue buckram. This sturdy library bound collection will be available in limited quantities, so order yours now.

The SoftSide Deluxe Edition — $75.00
Send a check or use your MasterCharge/VISA credit card.
To order write: **SoftSide Publications**
6 South Street
Milford, NH 03055

Hardware Corner

by Edward E. Umlor

That time has come around again — a new month and a new year. Last month we made ourselves a diskette, and now we are going to have to put it into something. Well, we are just going to have to talk about the disk drive itself first, then we can put our diskette into it and do some formatting.

What are the basic requirements for a disk drive?

1. A main frame to mount all the goodies on.
2. A motor, spindle, and spindle clamp to grip the diskette and spin it inside the jacket.
3. A holder for the jacket with write-protect sensor.
4. A precision read/write head positioning system.
5. A sensor system for index/sector holes.
6. A disk drive motor speed-control electronics.
7. A head-positioning electronics with a track 00 detector.
8. A data buffer and input control signal buffering with termination if required.
9. A data read/write control with data transfer from and to diskette.

I have probably broken this down a little further than some think is necessary, but I am sure there are some who have never seen a disk drive. What a delight they have in store, when they finally set the cassette aside for the (almost) real-time data transfer of the disk system. I will not be able to cover DOSs (disk operating systems) this month, but let me describe and explain the parts of a disk drive listed above.

1. The main frame is made out of two materials that I know of as of this writing. The first is cast aluminum from precision molds — cooled, and tempered under tight controls to reduce as much as possible any internal stresses. Any warp at all can cause the diskette and head to never come into proper alignment. The same process is applied to the high-impact, fiber-filled, and plastic main frame. (TEAC is the only one using plastic that I am aware of although all door assemblies for floppies are plastic — see Figure 1.) Aluminum is used by the rest of the drive manufacturers. They feel it is more rigid, less susceptible to temperature extremes, and stays true over the longest period of time. However, you don't want to drop, tighten into a misaligned chassis, or subject an aluminum frame to torsional stress. It can take a permanent set and thus be rendered useless. My experience with the plastic main frame, so far, has been a recovery back to its original alignment after the stress has been removed. I have one that a UPS truck tried to back over (the face was badly fractured) and it is still working fine.

2. The first things we will mount onto the main frame are the disk drive motor and spindle (the spindle clamp is actually mounted on the door assembly.) Take a look at Figure 2 and you will see the flywheel, drive motor, and drive belt connecting the two. The spindle is made of the following parts from the inside to outside: hub locked onto drive shaft, sealed ball bearing as shaft enters frame, a spring, exit ball bearing, and then the flywheel locked to the shaft with a center screw. On the flywheel are two strobe light strips (one for 50 Hz power and one for 60 Hz power). You can adjust the speed of your disk drive with a fluorescent light.

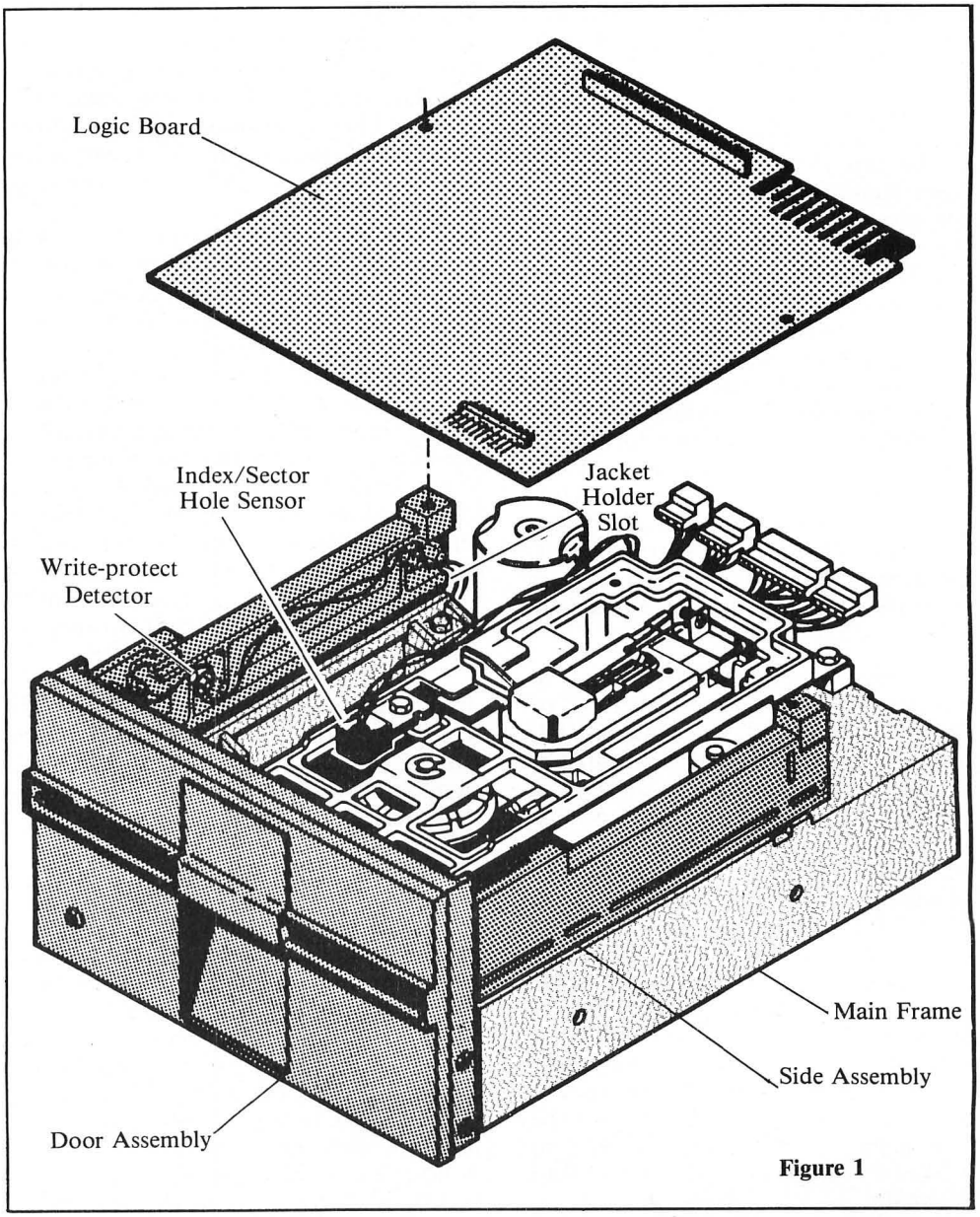


Figure 1

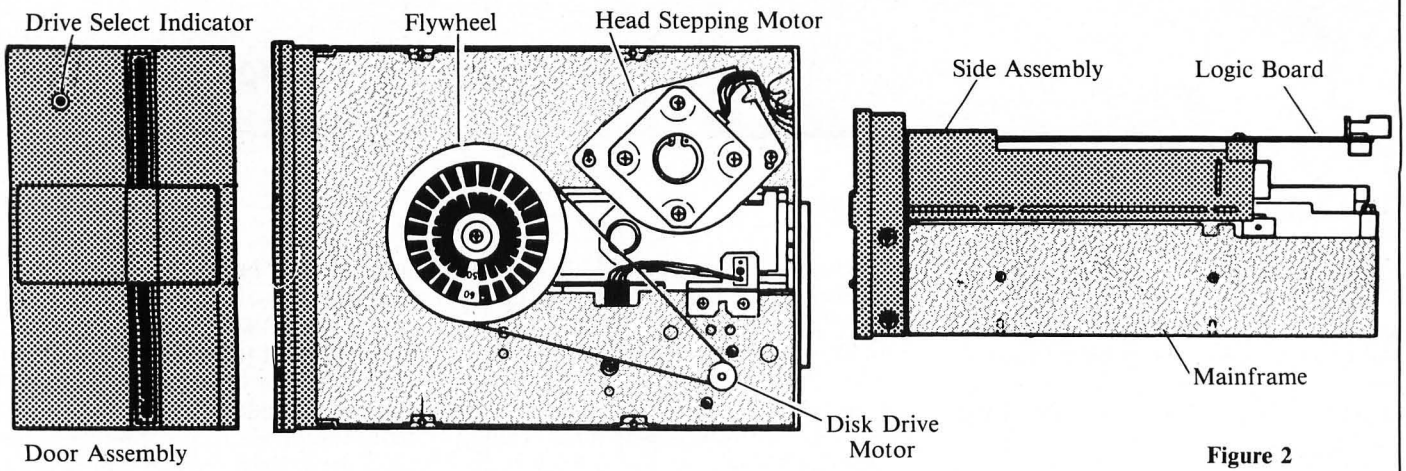


Figure 2

Remember that a fluorescent light varies its light output with the actual flow of electrons through it! An incandescent light doesn't nor does the sun. The fluorescent light is the **only** one that can be used as a strobe light to adjust your drive speed. The correct rotational speed for most 5 1/4 inch floppies is 300 RPM.

3. The jacket holder is usually two slots that line up with the disk opening in the face. These are a part of the main frame mold in most cases, but can also be plastic sides mounted on the main frame. On the proper side, you will find the write-protect detector. There are two types being used. The first is a lever-activated microswitch (the tip of the lever drops into the notch on the diskette). The second is a LED (light-emitting diode) and LDD (light-detecting diode) pair located to shine through the open area of the notch on the diskette. When the switch is not activated or light is detected, you can write to the disk without problems. Cover up that notch and when you try to write to the disk you will get an error message.

4. There is more than one way to position the head on the diskette. The first is a taut band. The taut band is secured to one end of the head carriage assembly, wraps around the drive shaft of the positioning motor, and on to be secured to the other end of the head carriage. One side of the band is slotted to allow the other side to pass through. Aligning the head on the taut band system is done by rotating the motor itself or by shifting the head carriage back and forth. The second way to position the head is by the wormscrew method. The head carriage is moved by a tab riding in a spiral groove cut into the shaft connected to the motor. The alignment of the head is accomplished by rotating the motor and thus the shaft.

5. This is a short one. The index/sector hole sensor has been a LDD/LED pair on all the drives I've worked on so far. The real necessity of these different sensors, etc., will be covered when we finally get around to formatting a disk.

6. The disk drive motor speed control is very important to good consistent data transfer. It can be a separate board (TEAC, Tandon, CDC, and others) or a part of the main control and logic board (MPI and some others). It consists of a voltage regulator (some are adjustable) and a second POT (potentiometer) for adjusting the actual voltage or current supplied to the drive motor. Within the drive motor is a speed detector to feed the actual speed back to the control board. The control board will then supply more or less voltage/current depending on the feedback to bring the motor back to correct speed. If your system does not include a data separator that is external to the disk control chip, then the normal tolerance in speed that you can stand without lost data is between one and two percent (3 to 6 RPM).

7. The head positioning control is driven internally from an external control pulse. This is under the direct control of the DOS and consists of two signals, head direction level and head step pulse. All the electronics in the disk drive are activated by the disk select level. This select level also activates the head load relay (actually a solenoid) that pulls the second head or pressure pad against the diskette. This forces the diskette into contact with the head. All head positions are SOFTWARE controlled. The head step pulse from outside the drive is translated into a specific number of degrees of rotation by the head step motor in conjunction with the internal drive logic. A 40-track drive steps approximately 1/40th inch track to track, and the 80-track drive steps approximately 1/80th inch track to track. Yes, only about a one-inch band of the diskette is actually used for data storage. Almost seems like a waste, doesn't it?

8. Data transfer to and from the disk drive is carried out in serial form. By nature all magnetic media are serial. The large tape transports that accept data in parallel form actually have nine read/write heads called channels to accomplish the job. You can see why they are very expensive. This conversion is accomplished within the disk controller itself. The disk control chip is what really ties the whole system together and we will discuss it right after the next section. The data are stored as two frequencies on the disk, and are called FM (frequency modulation). One frequency denotes a 0 and the other a 1. For double-density operation, it is called MFM (modified frequency modulation) and two higher frequencies are used. In the early days many disk drives were produced with read/write heads that could not handle the higher frequencies of the MFM format, but today all drives being built will handle the MFM. I don't think the industry thought that the double-density format would be so popular with the public.

9. Here again is an area that responds to the influence of the outside world. It is the software that tells the drive to read data or write data to disk. The R/W signal controls this function and either turns on the internal read circuits or turns on the internal write circuits. Since this is only one signal, its level determines the read condition or the write condition. The write condition turns off the data-sense amplifiers and allows signal to be supplied from the logic board to the head (now performing as a write head). In the read mode, the write circuitry is disabled and the head is connected to the data-

sense amplifiers. As the disk spins past the head, the magnetized particles induce a small voltage in the head. These small voltages are detected and amplified by the sense amplifiers and we have recovered the data put on the disk.

Well, that just about covers the drive itself. Any of these disk drives can be used on soft-sectored systems or hard-sectored systems. If you look at Figure 3, you will see how the data is organized (top) for soft-sector and hard-sector (bottom).

I am going to cover the formats, etc., as they apply to my Model I. The number of sectors differs from manufacturer to manufacturer, and sometimes one will change the number of sectors from one revision of a DOS to another. Whatever brand of computer you are using, the same principles apply. Only the method of implementation changes. When looking at the formats in Figure 3, it becomes very clear what happens if the rotational speed gets too fast: The end of the track's last sector is cut off or overrides the beginning of the first sector. It is usually the former as the index sensor will cut off the write operation and cause an error message to be displayed.

The disk control chip is the heart of the whole system. It does the timing, data transfer, read/write control, FM/MFM format, and general housekeeping for the disk. Let's say you have just told the computer to format your diskette (data disk). First the DOS tells the controller to seek track 00, so the controller sends out the select signal, starts the disk motor, loads the head, selects head direction, starts stepping the head, and checks track 00 sensor (a sensor similar to the write-protect) after each step. When it detects track 00, it looks for any correctly formatted sectors. If it finds the track already formatted, it lets you know about it through the DOS. This time we are clean. In the following, each of the GAP numbers represents a blank area of a different size and is expressed by number of characters wide (the number of characters that could be stored in that space). Now the DOS waits for the index hole to be detected and it sets GAP1 - ID Sector 1 - GAP2 - Fills data area with a specific sequence of characters - GAP3 - ID Sector 2 - GAP2 - Fills data area with same as sector 1 - GAP3 . . . etc. to the end of track (sector 10 for single-density or sector 18 for double-density) - GAP4. This format is the upper one in Figure 3 and is the soft-sectored format. Timing is all-important to the soft-sectored disk system.

The hard-sectored system is less critical of internal timing as each sector is initiated by a narrow pulse from the index sensor. If you look at the holes in a hard-sectored disk you will find the index hole is larger in diameter than the sector holes. Here we wait for the wide pulse from the index sensor, then we wait for the first narrow pulse, write sector 1 preamble, write the specified test data, write postamble, wait for narrow pulse, write preamble, test data, postamble, etc., for ten sectors or 16 sectors (the only two hard 5 1/4 formats I know of). This is the lower format in Figure 3.

I did not cover verifying the formatted track above as both the hard-sector and soft-sector systems use the same two methods of doing so. The most popular way is to verify the formatting of the track immediately on the next revolution of the diskette. The second method is to format the whole diskette and then verify the formatted tracks in reverse order or return to track 00 and verify in the forward direction. We have now covered the diskette, disk drive, and formatted a diskette. The only thing now is single/double density. If you were sharp, you already know the answer to that one now.

Single-density is cutting your disk into ten sectors for both hard- and soft-sectored systems. When we upgrade the DOS in a hard system to handle 16 sectors, we are using their double format. We increase our data storage by a factor of 1.6 as well as speed of data transfer. The disk is still rotating at 300 RPM. On my Model I with the Doubler, I now get 18 sectors per track. This is an increase of 1.8 in capacity and speed. Simple, isn't it?

Well, that is about all for now. Next month I will try to cover several different types of drives and what their differences are. I am a bench type and have not used very many of the different DOSs that are available, but I will try to recuss a little about the ones I have a noddin' acquaintance with.

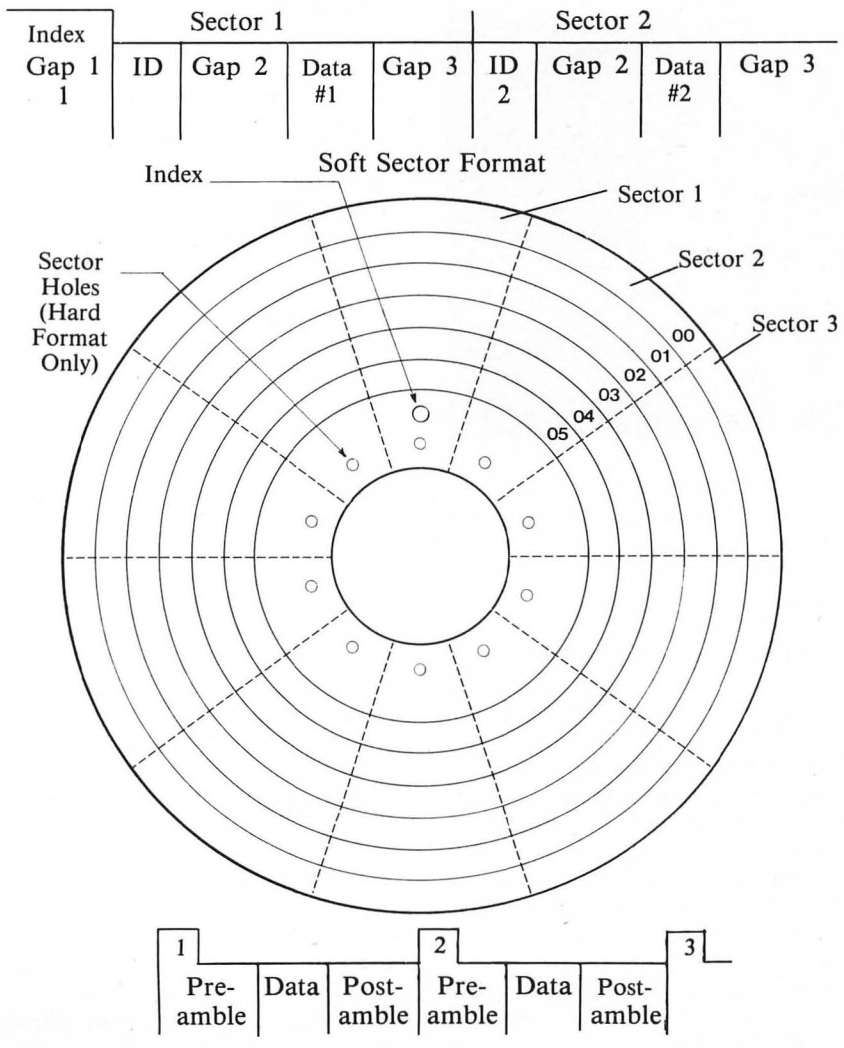


Figure 3

EXPERIMENTS IN ARTIFICIAL INTELLIGENCE FOR SMALL COMPUTERS

by John Krutch (Howard Sams) Suggested retail price: $6.95.

Reviewed by Luigi Bisceglia

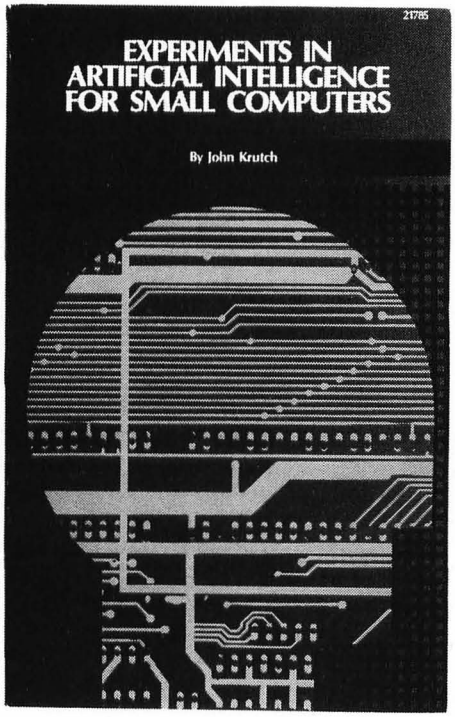
What a good idea: introduce the elementary concepts of artificial intelligence (AI) by using programming experiments in TRS-80® BASIC. The table of contents lists its potential: Chapter One briefly describes the scope and problems of AI. Chapter Two uses game-playing to describe trees, minimaxing, and the alpha-beta algorithm. Chapter Three discusses problem solving. Chapter Four describes programs that "reason" and "converse" syllogistically. Chapter Five talks about computer-created verse. Chapter Six continues that idea into computer-generated text. And Chapter Seven discusses psuedo-natural language responsiveness, i.e., *Eliza*.

However, the book's arrangement is a clue to its failure. Computer haiku and *Eliza* are not the penultimate state of computer intelligence. Such programs mimic aspects of intelligence but don't lead to it. Any number of machines could randomly select words from a limited, keyed vocabulary and "create" so-called haiku. In a computer program, the onerous task is entering the vocabulary; Krutch's is no different.

Chapter Two's discussion of trees was my reason for purchasing the book. As a reasonably intelligent person, I have a theoretical appreciation for AI, but technical discussions put my perceptive faculties to sleep. In this chapter, Krutch described the minimax tree search so simply that I could not fail to understand and then he went on to clarify pruning. The discussions threw light where once was darkness, but were limited to descriptions of what an algorithm does, not how it works.

And right there on page 20, the book lost its focus and direction: "In the interest of brevity CHECKERS does not incorporate a move-by-move tree generation procedure as previously described." Nothing in the rest of the book is worth omitting an in-depth example of that procedure. Once the author failed to further our understanding of tree searching, he also wandered away from the promised experiments toward computer intelligence.

Even *Sargon*, for example, with its impressive list of victories remains a "dumb" program. That is not meant



in any way to diminish the Spracklens' achievement, but *Sargon* doesn't learn except as they do; each contest is a tour-de-force unrelated to the past or future; it's what was once called an idiot-savant.

All of Krutch's examples suffer from that malaise. Every collection since *What Do You Do After You Hit Return* has included similar simplistic versions of haiku, checkers, and *Eliza*. Why didn't Krutch present real experiments toward artificial intelligence even at the elementary level of BASIC programming? In his haiku program, why not an algorithm to request help in developing vocabulary — and then let the program write poetry?

Krutch failed to live up to his concept. The book amounts to a collection of seven inappreciable BASIC programs that are not worth $6.95 plus tax. I am excited by the hope that we might create an alternate life form or intelligence. Maybe I won't live on the moon or colonize deep space, but I can participate in this adventure close to home. Unfortunately, Krutch's book will not help any of us share the adventure.

GRAPHICS EDITOR AND PROGRAMMER

by Bill Mason (J.F. Consulting) System Requirements: 16K tape-based or 32K disk-based TRS-80® Model I or III. Suggested retail price: cassette — $25.99

Reviewed by Margaret M. Grothman

Of all TRS-80® programs that have passed my way, none has exceeded my expectations as this one has, *Graphics Editor and Programmer*, by Bill Mason.

With *GEAP*, you can create a screen display consisting of both graphics and text, generate the BASIC program to recreate the display, and save it to tape or disk for use in your own programs. Other programs advertise the same services — but *GEAP* has so many extras, and saves so much of the programmer's precious time, that it deserves special recognition.

I sent for *GEAP* in response to a direct-mail solicitation. Having been burned before, I was skeptical, but willing to risk the money on the chance that *GEAP* would perform as claimed. It has surpassed all hopes.

Like any complex program, *GEAP* takes time to learn. The documentation is not elaborate, but it is very clear, and arranged in a way that allows you to begin using the program right away. *GEAP* has four modes: regular, print, keypad, and designate. Elementary commands in each mode are introduced before going into the frills.

Regular and Print Modes

The regular mode allows you to draw on the screen by holding down an arrow key and the shift key. Moving an unshifted arrow erases or moves the cursor without drawing. A picture can be saved at any time by pressing ENTER, and can be retrieved by press-

ing the spacebar. CLEAR clears the screen. If you want words on your display, you go to the print mode via the asterisk key.

When you are satisfied with the display, you call a menu for instructions to create the program. You can specify compressed or uncompressed code. Compressed code, like packed strings, cannot be edited, but uses much less memory than uncompressed code. Unless you anticipate a need to edit the picture later, you would want to use the compressed form.

After the program has been made, you can RUN it, LIST it to the screen or printer, or SAVE it on tape or disk. While you do this, *GEAP* is still in memory, and you may return to it to make more displays.

Keypad Mode

In keypad mode, you can:

(1) Draw a horizontal line through the cursor by pressing the spacebar.

(2) Press ENTER to display the print location of the cursor. This is useful for marking a spot for printing program variables later.

(3) Move the entire screen in any direction. A very handy command.

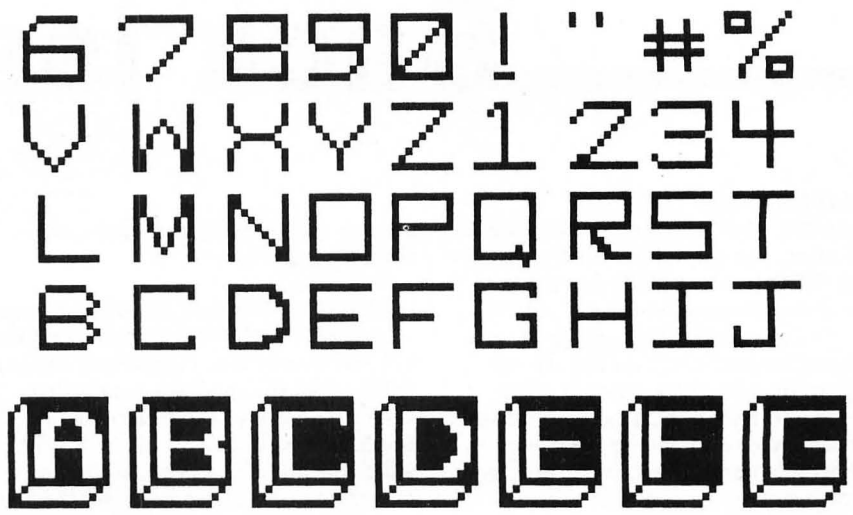
(4) Use six keys (Q, W, A, S, Z, X) as a graphics keypad to draw any of the 63 graphics patterns. Once you have drawn a pattern, you can use the shift and arrow keys to draw a row of that pattern. You could use this feature, for example, to draw with a square cursor, instead of the usual rectangular, single pixel cursor. This feature is also handy for drawing fancy borders around a display.

Designate Mode

This mode is the most complicated of the four *GEAP* modes, but also the most fun. In the designate mode, you can mark off a part of the picture, and then manipulate it in various ways. You can fill it with a graphics character of your choice or you can reverse black and white. Once you have "designated" a figure, you can move it, duplicate it on other parts of the screen, rotate it, magnify it, or produce a mirror image of it. You can even change its dimensions. One of the most intriguing things that can be done with the designated figure is to use it as a paint brush. By holding down the slash key, the shift key, and an arrow key simultaneously, you can make a row of figures, each one superimposed on the last, but offset by one print location. You could draw a pile of boxes, a row of houses, or a fence.

The designated figure can be stored in a string with a name of your choosing. The instructions to create the string also create a matching string to erase your figure. This is very useful for programs which use the same figure in different locations and at different times, for example, game tokens. The code containing the strings are in compressed form and can't be edited.

Incidentally, merging *GEAP* displays or strings into another program is a simple POKE and PEEK procedure. If you don't already know how to merge two program segments, the instructions are given in an appendix to *GEAP*.



Above: Samples of the over-sized "Super Graphics" character sets made with two GEAP Expansion Modules. All four Expansion Modules which contain different enhancements to the GEAP program are available together for $16.99.

Other Features

You can LOAD a screen from another program for editing and coding with *GEAP*. I have used this feature to recode program segments from another program into the compressed code to reserve space.

Two displays may be merged. A display can also be programmed to appear in stages by inserting time delays.

If the program you are writing requires input from the user, a quote mark will result in a prompt at that point when the display is coded.

And last, if you can think of any other commands that you would like, instructions are provided for expanding *GEAP*.

Flaws

Can *GEAP* do everything? Well, no. It doesn't carry out trash and wash the dishes while you are programming. Otherwise, it is hard to find fault.

One minor problem has to do with the spacebar, which is used to call back pictures that have been saved. Several times I accidentally brushed against the spacebar, losing the picture I was working on. To prevent the loss of your hard work, it is a good idea to save the screen frequently. There is an alternate temporary screen storage which can also be used.

Summary

I use my TRS-80® for educational program development. In pre-*GEAP* days, I made the video displays for my programs in a painstaking manner. I used a program which allowed me to draw pictures on the screen, then LIST the print locations to the printer. After the locations were known, I assembled them manually into strings or into data statements to be POKEd into the video memory. Then the words had to be imposed onto the graphics display by means of PRINT@ statements. This was all very time consuming. With *GEAP*, I can do this work in about one-fourth of the time it used to take.

But my enthusiasm is based on more than just a good program. I used *GEAP* while it was being test-marketed. During this time, my correspondence with the author, Bill Mason, and with J. F. Consulting, which markets the program, was rewarded by a high level of concern and willingness to help. These are people who obviously care about the quality of their product and the satisfaction of their customers. They deserve to be successful.

Back Issues from

SOFTSIDES FROM THE PAST

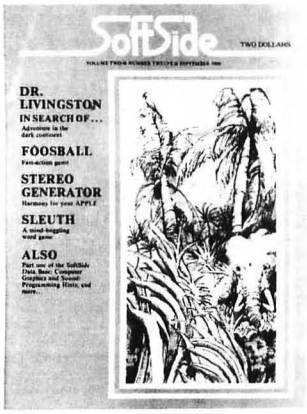
If you like what this issue of **SoftSide** has to offer, you should see what's waiting for you in **SoftSide** back issues! You may feel that you've missed out on many of our programs that appeared before you became a subscriber. It's not too late to do something about it. You won't have to miss a thing because we still have back issues available to complete your **SoftSide** library! But, order now as some of our more popular issues are already out of stock and others are dwindling quickly.

Listed below are all of our past issues with their FEATURE programs and the systems they're for. For a more complete index of all the programs and articles offered in each of the back issues of **SoftSide** please refer to the May, 1981, issue. Each issue costs $3.50 for the magazine only. Those issues marked with an asterisk are also available with the program on cassette for $9.95 or on disk for $14.95 (except DV issues).

The enhanced Disk Versions (DV) contain an extra program for each system. The TRS-80® DV began with the September, 1981, issue. The Apple DV began October, 1981, and the Atari DV started in November, 1981. Each enhanced DV costs $19.95.

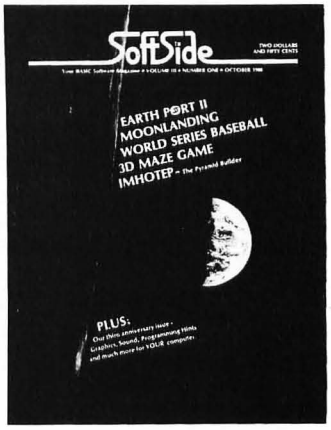
***August 1980**

"Caribbean Cruising" — Apple
"Master's Golf" — ATARI®
"Sailplane" — TRS-80®



***September 1980**

Magazine not available
Media edition only
"Developing Data Base"
All Systems
"Stereo Generator" — Apple
"Barricade" — ATARI®
"Concentration II" — TRS-80®



***October 1980**

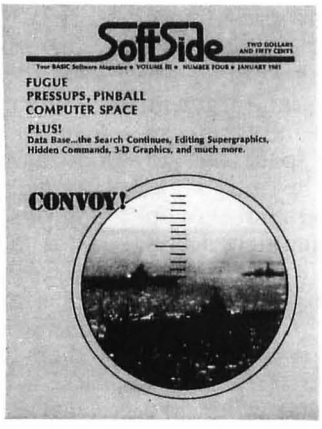
"Developing Data Base II"
All Systems
"Moonlanding" — Apple
"World Series" — ATARI®
"Earth Port II" — TRS-80®

***November 1980**

"Developing Data Base III"
All Systems
"Collision" — Apple
"Trench" — ATARI®
"Kriegspiel" — TRS-80®

***December 1980**

"Developing Data Base IV"
All Systems
"Baseball" — Apple
"Speedello" — ATARI®
"Kidnapped" — TRS-80®



***January 1981**

"Developing Data Base V"
All Systems
"Convoy" — Apple
and TRS-80®
"Angle Cannon" — ATARI®
"Ship Destroyer" — TRS-80®

***February 1981**

"Developing Data Base VI"
All Systems
"Miner" — All Systems
"Mini-Golf" — ATARI®
and TRS-80®
"Long Distance" — TRS-80®

***March 1981**

"Developing Data Base VII" — All Systems
"Strategy Strike" — Apple
and TRS-80®
"Flags" — ATARI®
"Volcano" — TRS-80®

SoftSide™

Use the bind-in card in this issue to order or send a list of the back issues you'd like, with payment of $3.50 per magazine.

SoftSide Publications
515 Abbott Drive
Broomall, PA 19008

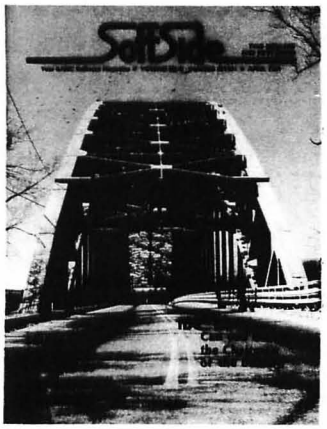
For the magazine/media combination, send $9.95 per cassette and magazine, $14.95 per disk and magazine, or $19.95 per DV and magazine to:

SoftSide Publications
6 South Street
Milford, NH 03055

***December 1981**

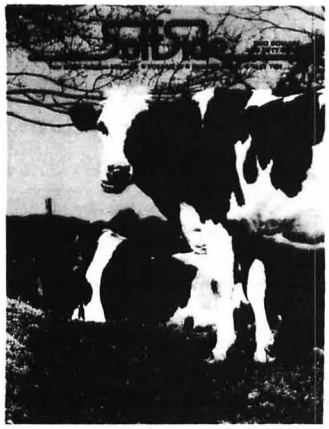
"Titan" — All Systems
"Aircraft Commander" — Apple
"Developing Data Base"—ATARI®
"Electronics Assistant"—TRS-80®

Enhanced Disk Version
"Bobsledding" — Apple
"Survive" — ATARI®
"Konane" — TRS-80®



***April 1981**

"Battle At Sea" — Apple
"Convoy" — ATARI®
"Dominoes" — TRS-80®



***May 1981**

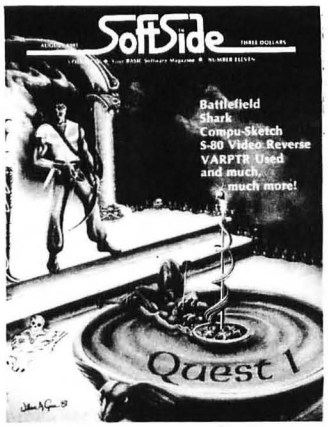
"Galaxia" — Apple
"Dodge" — ATARI®
"Orienteering At Jacque's Coulee" — TRS-80®

***June 1981**

"Old Glory" — All Systems
"Word-Search Puzzle Generator" — All Systems
"Anallist" — TRS-80®

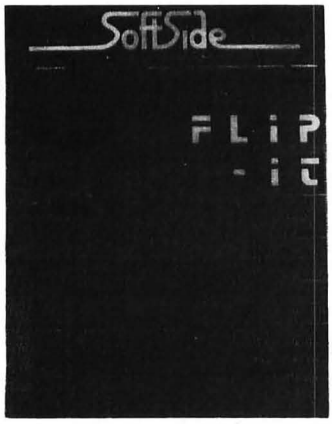
***July 1981**

"Chemistry Drill" All Systems
"Kidnapped" — Apple and ATARI®
"Magic Paper Calculator" — TRS-80®



***August 1981**

Magazine not available
Media edition only
"Quest 1" — All Systems
"Battlefield" — All Systems
"Compu-Sketch" — TRS-80®



***September 1981**

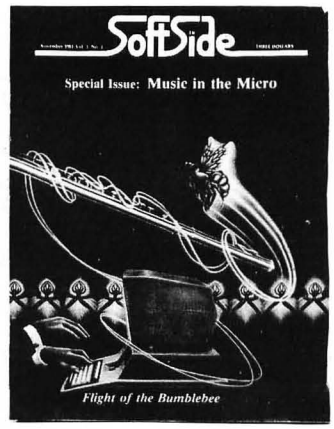
Magazine not available
Media edition only
"Flip-It" — All Systems
"Word Challenge" — All Systems
"Exterminate" — TRS-80®

Enhanced Disk Version
"NewBASIC" — TRS-80®

***October 1981**

"Leyte" — All Systems
"Developing Data Base"—Apple
"Character Generator" ATARI®
"Envyrn™" — TRS-80®

Enhanced Disk Versions
"Super Dairy Farming" — Apple
"Gameplay" — TRS-80®



***November 1981**

"Flight of the Bumblebee" All Systems
"Music Machine" — Apple
"Music Programmer"—ATARI®
"Music Editor" — TRS-80®

Enhanced Disk Versions
"National Anthems" — Apple
'Volleyball" — ATARI®
"Mean Checkers Machine" TRS-80®

SoftSide Selections Terms & Conditions

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.

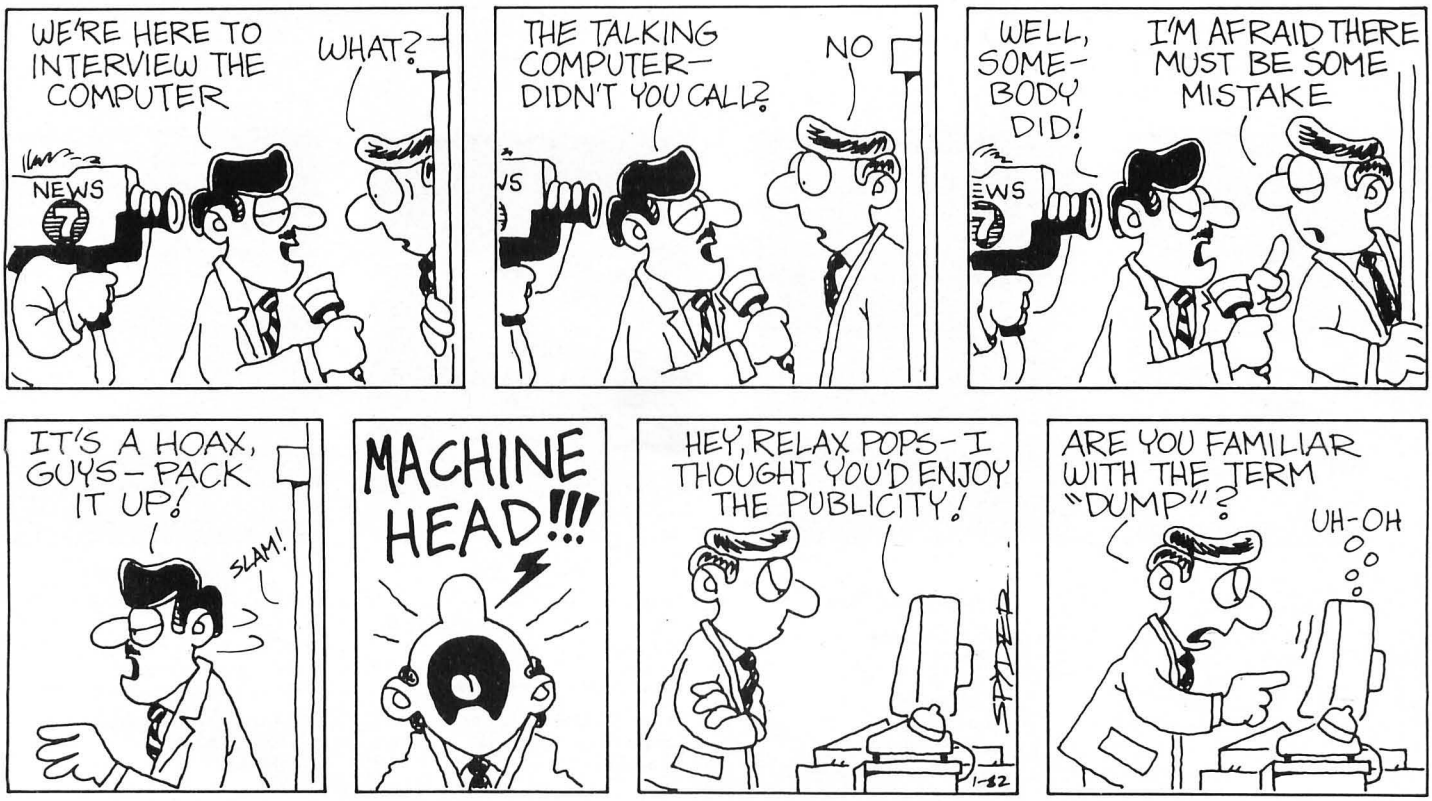
Liability
All software is sold on an as-is basis. **SoftSide** assumes no liability for loss or damage caused or alleged to be caused directly or indirectly by products sold or exchanged by them or their distributors, including, but not limited to, any interruption in service, loss of business or anticipatory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

Advertiser's Index

MACHINE HEAD

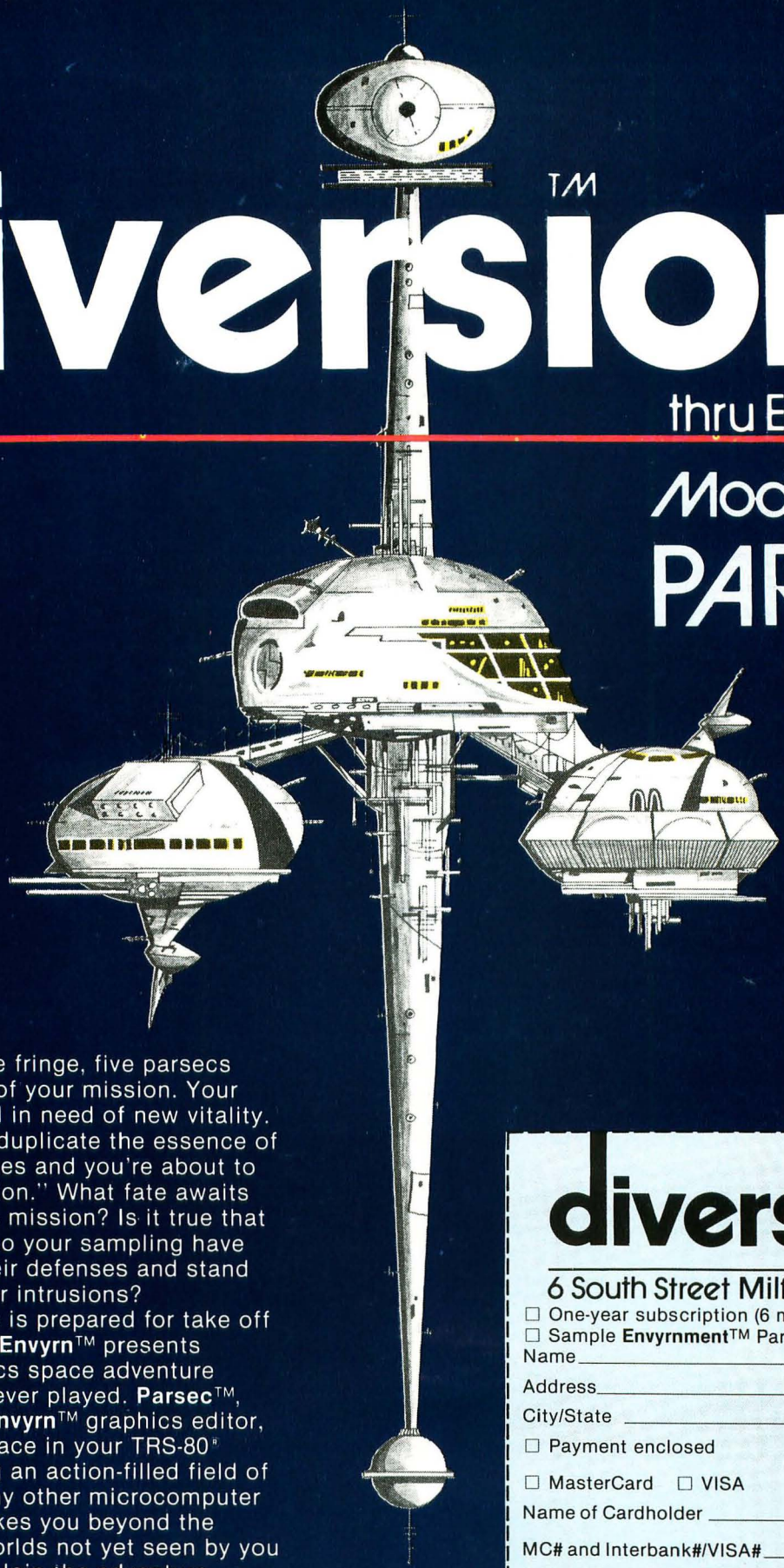
BY SPYDER



diversions™

thru Envyrn

Module One
PARSEC™



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

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

diversions™
thru Envyrn

6 South Street Milford, NH 03055

☐ One-year subscription (6 modules) — $60.00
☐ Sample **Envyrnment**™ Parsec™ only — $20.00

Name _____

Address _____

City/State _____ Zip _____

☐ Payment enclosed

☐ MasterCard ☐ VISA Exp.Date _____

Name of Cardholder _____

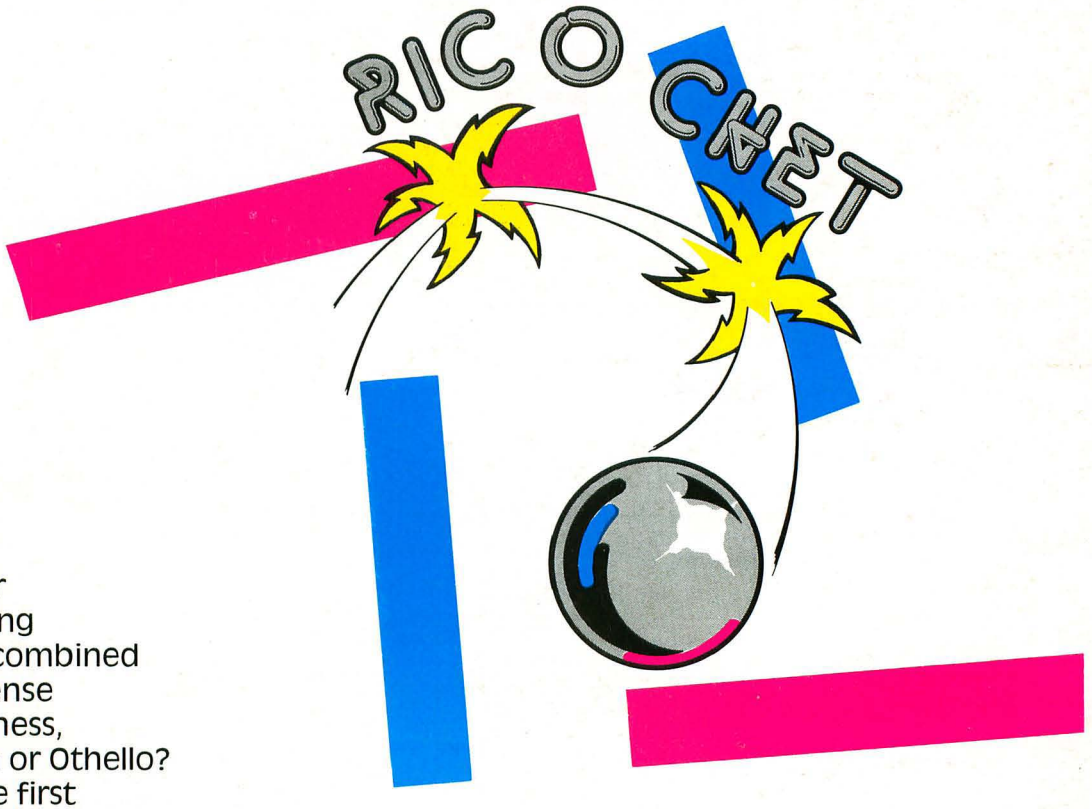
MC# and Interbank#/VISA# _____

Signature _____

To order use the card provided in this issue or send in this order blank.
I own a 48K TRS-80® with Disk ☐ Model I ☐ Model III

TRS-80 is a registered trademark of Tandy Corporation

It's Here! The Computer Strategy Game with Bounce!



For:
Apple
Atari
TRS-80

Have you ever seen an exciting action game combined with the intense strategy of chess, backgammon or Othello? Ricochet...the first abstract strategy game designed **exclusively** for the computer owner... is both. And loads of fun.

You maneuver your blocks, both to protect your own goal from attack **and** to hit your opponent's goal. Two launchers to fire. Your shots ricochet off the blocks, earning you points on the way to their targets. It's twice as challenging because the position changes with both your own and your opponent's moves and shots.

You don't have to play alone, either. Play against any one of four different opponents (each a different personality) inside your computer, or against another human.

And Ricochet is truly competitive...if you want it to be. A "smart clock" lets you put more pressure on your opponent by forcing him to play faster than you. But you've got to win two out of three (or three out of five) games to claim victory. Your computer rates you after each match, so you can compare your mastery of the game with that of other players—perfect for tournament play. So perfect that MIND TOYS and Automated Simulations are sponsoring the first national Ricochet tournament. See your local dealer to find out how you can become a regional or national champion.

Price $19.95

Another Mind Toy From Automated Simulations

P.O. Box 4247, Mountain View, CA 94040.

*APPLE, ATARI and TRS-80 are trademarks of Apple Computer, Inc., Atari, Inc. and Tandy Corp., respectively.

mind toys