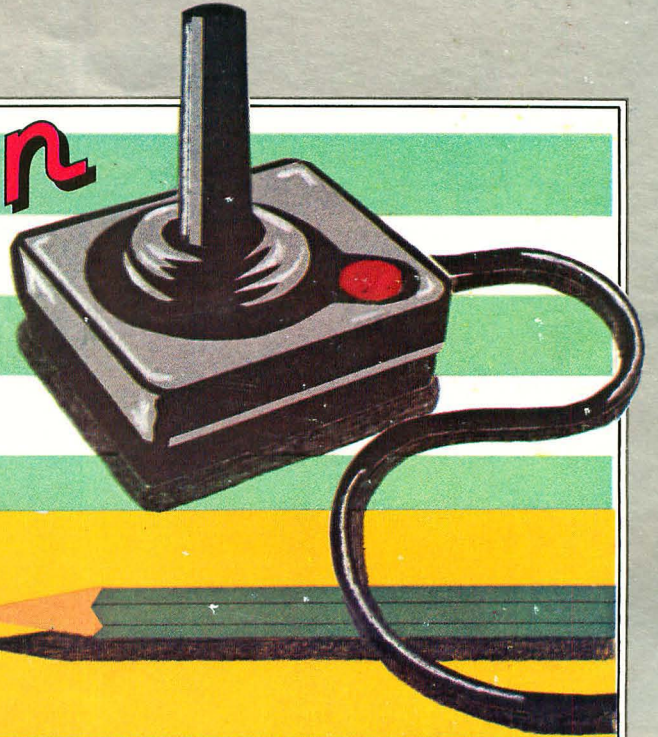


Computer Awareness in Education



C.A.T.S.

*Computer Assisted
Testing System*

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

14112 G.B.

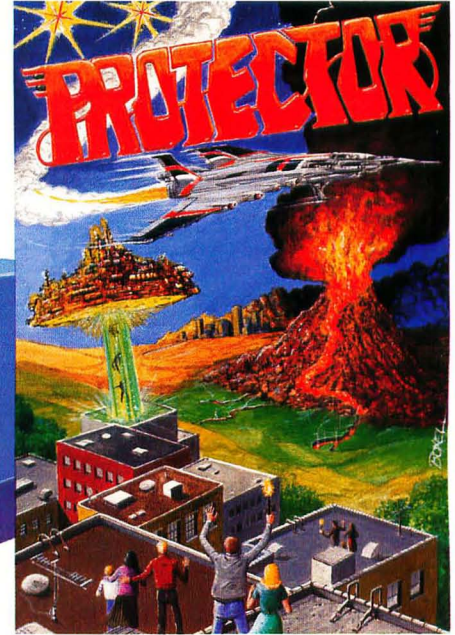
L. Wood
1982

**THEY LAUGHED AT ME WHEN I SAT DOWN AT THE KEYBOARD . . .
BUT OH, WHEN I BEGAN TO PLAY! HERE'S WHAT THEY SAID:**



BPL-ATON I — "I've had to replace three electro-universal wristcouplers since I got it."

PROTECTOR.



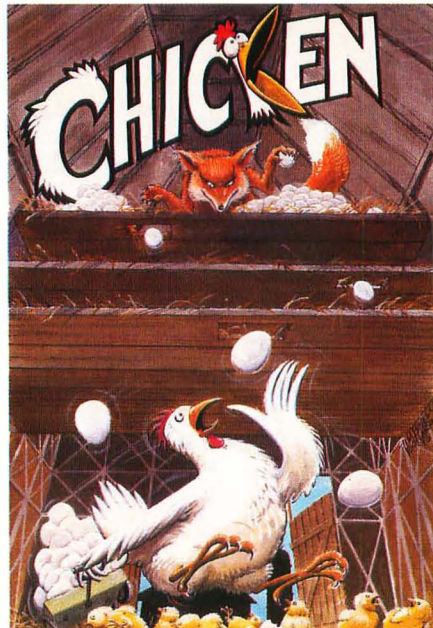
CORGUS I — "Even with a class II embryonic neurolifter, it is disgustingly difficult to play."

SLIME.



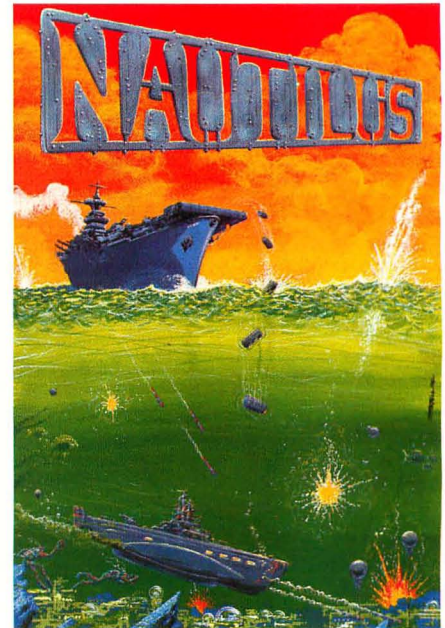
ADM-MARK II+ — "More fun than kicking the servonic katdroid."

CHICKEN.



ELF-X2 — "Can I be a submarine when I grow up?"

NAUTILUS.



NAUTILUS, PROTECTOR, CHICKEN AND SLIME ARE ALL TRADEMARKS OF SYNAPSE SOFTWARE.

synapse
SOFTWARE

820 Coventry Rd.
Kensington, CA 94707
(415) 527-7751

FOR THE ATARI* HOME COMPUTER.

*ATARI IS A REGISTERED TRADEMARK OF ATARI, INC.

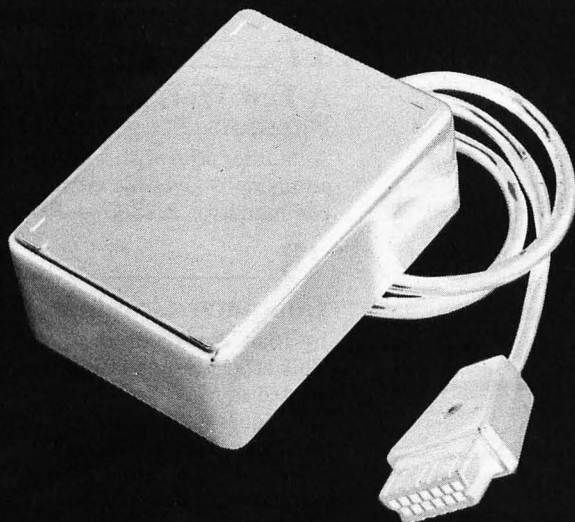
ATARI SAYS ITS FIRST WORD

WITH A VOICE BOX BY THE ALIEN GROUP!

THE ALIEN GROUP has emerged from the underground, daring to offer a full-featured speech synthesis system that is flexible, low in cost, and needs no accessory devices. No Interface, Cables, Speaker, Amp, or External Power Required! **VOICE BOX** has been designed and programmed by Atari users to become the integral voice of a 400 or 800 computer. Simply plugged into the serial port, **VOICE BOX** automatically routes all speech into the speaker of your television monitor. With the menu-driven operating system supplied, you'll be creating original, intelligible speech within moments after loading disk or cassette. No lengthy or obscure instructions to wade through.

The system includes a dictionary which translates typed text into **VOICE BOX's** phonetic language. The dictionary can be expanded to include as many as 5,000 words of your own custom vocabulary. Unlimited speech can be produced by straightforward phonetic definition at any time, even if the dictionary should be full.

The speech routines can be called from other programs for any purpose you can conceive. Here are a few suggested uses:



VOICE BOX
Speech Synthesizer

SOUND EFFECTS

- Access 64 phonemes at any of 4 pitch levels to add filtered, contoured sound to the Atari audio repertoire.

GAMES

- Program aliens to hiss threats, moan when destroyed.
- Devise weird, non-human tongues for dungeon dwellers.
- Insert cryptic spoken clues in maze games.

COMPUTER OPERATION

- Code verbal prompts and error messages that command attention and leave the current display intact.

EDUCATION

- Gain an introduction to the principles of phonetics.
- Learn touch typing through spoken feedback from the keyboard.

In addition, the *Random Sentence Generator* included in the operating system, which prints and speaks endlessly startling, amusing, even poetic combinations of words supplied by the user, helps teach school children to identify parts of speech and recognize a variety of sentence structures.

A minimum of 16K RAM is required by the operating system. Either disk or cassette includes both 16K and 32K versions. Try **VOICE BOX** for up to 10 days, and if it isn't the finest value you've ever seen in a computer peripheral, the most challenging and provocative addition you've ever made to your system, return it in its original condition for a full refund.

When ordering specify disk- or cassette-based operating system, and enclose check or money-order for \$169, or state your **VISA** or **MASTERCARD** number. Send mail orders to: **THE ALIEN GROUP**

27 West 23rd Street
Dept.
New York, New York 10010

or telephone orders from
10 AM to 6 PM New York time
(212) 924-5546

ALSO AVAILABLE AT LEADING COMPUTER STORES THROUGHOUT THE WORLD.

Atari is a registered trademark of Warner Communications.

SoftSide™

EDITOR
Randal L. Kottwitz

SOFTWARE EDITOR
Bill Kubeck

EDITORIAL STAFF
Rich Bouchard
Alan J. Zett

EDITORIAL ASSISTANT
Joyce Smith

CONTRIBUTING EDITORS
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Jon Voskuil

ART PRODUCTION MANAGER
Lynn Wood

PRODUCTION STAFF
Lynda Fedas
Denise Lafleur

PUBLISHER
John G. Grow

ASSOCIATE PUBLISHER
Nancy Lapointe

ADVERTISING MANAGER
Sue Rowland

CUSTOMER SERVICE
Cindy Schalk

DEALER SALES
Kathie Maloof
Irene Stanton

STAFF
Mary Edwards
Donna Jean
Steve Justus
Doris Miller

FOUNDER
Roger W. Robitaille, Sr

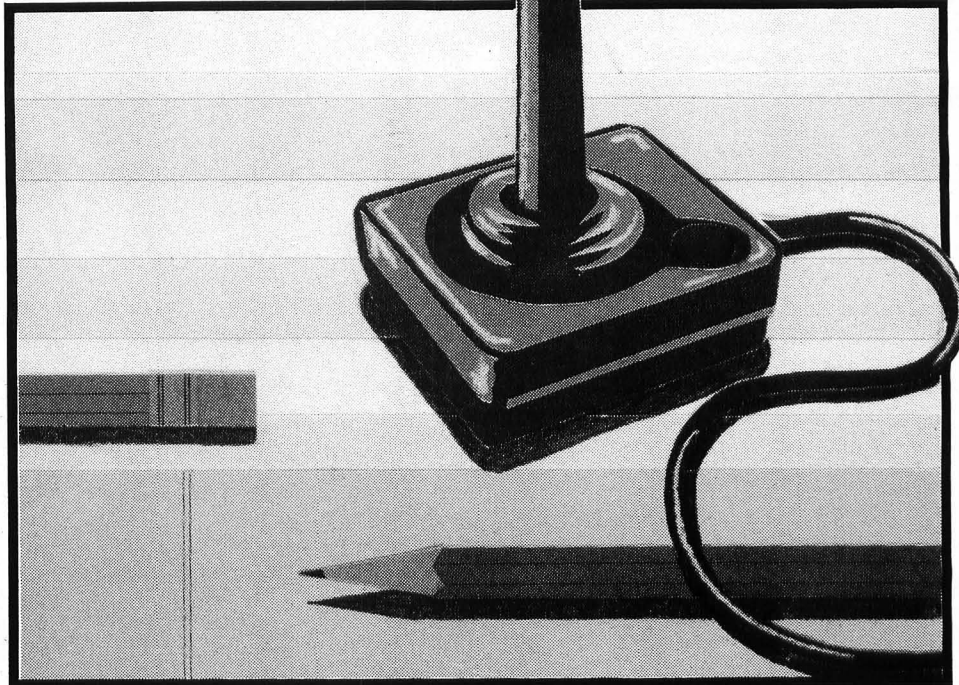
Photographs by Dean F. H. Macy

SoftSide Vol. 5, No. 10

SUBSCRIPTION INQUIRIES should be sent to *SoftSide* Publications, 100 Pine Street, Holmes, PA 19043. EDITORIAL AND ADVERTISING CORRESPONDENCE should be sent to *SoftSide* Publications, 6 South Street, Milford, NH 03055. Telephone (603) 673-0585.

SoftSide (ISSN 0274-8630) is published monthly by *SoftSide* Publications, Inc., 6 South Street, Milford, NH 03055. Telephone 603-673-0585. Printed at Lorell Press, Avon, MA. Second class postage paid at Milford, NH and at additional mailing offices. Subscription rates: USA, APO, FPO, and Canada, \$30/12 issues. First Class USA, Mexico, \$40/12 issues. Other foreign countries, \$62./12 issues. Media subscription rates: US, APO, FPO, Magazine and Cassette, \$75/12 issues. US Magazine and disk, \$125/12 issues. Canada, and Mexico, add \$20/12 issues. Other foreign add \$50/12 issues. All remittances must be in U.S. funds. Entire contents Copyright © *SoftSide* Publications, Inc., July, 1982. All rights reserved. POSTMASTER: Please send form 3579 to *SoftSide* Publications, 100 Pine Street, Holmes, PA 19043.

TRS-80®, Apple™, and ATARI® are registered trademarks of The Tandy Corporation, The Apple Computer Company, and Warner Communications, respectively.



Cover illustration by Lynn Wood

FRONT RUNNER

23

CATS

by Jon Voskuil
Translations by Alan J. Zett
The Computer-Assisted Testing System will turn your micro into a masterful testing machine. This time we bring you the input module and will complete this educational utility in the next issue.

FEATURES

11

Entertainment Tomorrow

by Fred D'Ignazio and Allen W. Wold
The authors continue their series on computer aided special effects in the film industry. This time — a fascinating scenario considering the movie theatres of the future.

14

My Side of the Page

by Lance Micklus
Lance has received some interesting response from readers of his review of the Modem I. In addition, he gives us his comments on supposed software rip-offs.

17

A Few Words From the Phantom Programmer

An author who wishes to remain anonymous gives his views on a new command for BASIC — PUNT.

18

Computer Graphics

by Joan Truckenbrod
This article is a continuation of the series started in May. Transformation techniques are explored in-depth, varying the rates of change during a transformation series.

20

Why Johnny Can't Program — Materials for Computer Literacy

by Dean F. Hayden Macy
Teachers are finding themselves in a quandry when choosing educational materials to teach computer awareness. The author gives an overview of the available resources and some helpful tips on how to utilize them.

96

Machine Head

by Spyder Webb

DEPARTMENTS

- 4 Editorial
- 6 Input/Output
- 8 Hints and Enhancements
- 10 Calendar
- 10 Bugs, Worms and Other
 - Undesirables
- 41 General Information
 - Concerning *SoftSide* Line Listings, SWAT and Media Versions
- 94 New Products
- 96 Advertisers Index

APPLE™/SIDE




- 42 **K-Byter**
SNAKE SCRAMBLE by Leonard Vincent
Race around the base of a tree, swallowing the fallen apples as you go. Be careful though, take a bite of your own body and you're a goner.
- 44 **Enhanced Disk Version**
APPLESOFT EXTENSIONS by Kerry Shetline
Here are some welcome and badly needed enhancements to the features of Applesoft, including RESTORE to line number and a true LINE INPUT.
 - Programs**
- 46 **PUZZLE JUMBLE** by Steve Faiella
Create your own custom puzzles, mix them up and try to put them back together again. It's not as easy as it might sound.
- 53 **QUIK FOLIO** by Rod Packer
This compact, extremely well-documented program allows the amateur investor to perform a simple analysis of his portfolio.
- 55 **Review**
FIREBIRD Review by Hartley G. Lesser

ATARI®/SIDE



- 58 **Program**
TUNEIN by William Morris and John Cope
A colorful implementation of "Simon Says," this sound

and graphics game is a joy of frustration to play. In addition, the authors have given us an interesting tutorial on complicated graphics without Machine Language.

- 62 **K-Byter**
PICTURES AT AN XIO-BITION
by David Suwala
Through ingenious use of the ATARI®'s XIO commands, this little gem creates paintings on your screen Mondrian would have been proud to sign.
 - 64 **Enhanced Disk Version**
MENU PLUS by Paul Marentette
Have you ever wished you could see a menu of your disk the moment you booted your system? Well, now it's possible, and so are many other extremely convenient features with this utility.
 - Reviews**
 - 66 **SIMULATED COMPUTER** Review by Craig Chamberlain
 - 68 **MICROSOFT BASIC**
Review by Sheldon Leemon
 - Article**
 - 73 **EXPLORING THE ATARI® FRONTIER**
by Alan J. Zett
This time we further explore the ATARI® display list. Through a detailed example, the author constructs an elaborate "Title Page" sequence.
-
- ## TRS-80®/SIDE
-
- 

- 78 **Enhanced Disk Version**
STARBASE GUNNER by David Hillard
Get ready for fast action in deep space. The enemy is attacking at high speed and you are the only one who can stop them!
- 82 **K-Byter**
PEGBOARD SOLITAIRE by Bernard Harford
The wood board and golf tees you used to build this game in your crafts class are no longer necessary. Put them away and play this game classic on your screen.
- 83 **Program**
DOTS by Charles E. Wooster
Remember the pencil and paper game the teacher always used to yell at you for playing when you should have been studying? Well, here it is on the screen of your micro. You can play against your friends or the computer. Watch it — no one at *SoftSide* has beat the computer yet!
- 91 **Review**
ZBASIC 2.0 Review by Tim Knight



A Cause Without A Rebel

by Randal L. Kottwitz

As the ever greater influx of technology and its effects are flooding our society, the educational system in the U.S. is facing a serious crisis. It is caught in a trap between making serious and costly overhauls in its very core approach to educating our children and having to drastically alter the way in which it spends its money. It faces this dilemma as a disenchanted public cheers the government's deep paring of educational budgets. The editorial staff of *SoftSide* also found themselves somewhat disenchanted as they explored the subject matter for this issue — Computer Awareness In Education. Our disenchantment came not from the lack of available informa-

tion, for it seems to abound in the pages of every type of publication considering the microcomputer field, but from the lack of *focus* from which that information is suffering.

Mention the subject of computer education in almost any polite conversation today, and there will be more than a murmur of positive support and even a few "It's about time they did something practical!"s. (There is a small organization being formed in California which would strongly disagree, but their approach to prohibiting computer education, considering the current state of society, is almost laughable.) But, get into a discussion of the methods by which such education should be implemented, and you'll find yourself in a bees' nest of disagreement and confusion. Indeed, the parents being faced with the issue find themselves confronting alterations in not only the method by which the "traditional" subjects are being taught, but a drastic change in the subject matter as well. Regrettably, the change is so dramatic, many teachers find themselves in a similar state of affairs. It's understandable that both parents and teachers who are products of an era in which the computer became "Big Brother" to us all, are somewhat confused. They must try to differentiate the positive effects of a teaching computer and the negative distraction the same machine can become to the study of rudimentary essentials if allowed to serve only as a game machine. (I herald the introduction of the movie, *TRON*, from Disney Studios this summer, but question what effects it may have on the general public's attitudes towards the entertainment side of computing. The "Big Brother" we all fear is certainly personified in the *Encom Corporation*.)

I would maintain that the main cause of the confusion which plagues computer awareness in education is the lack of a true "clearing house" of information concerning the field. There are

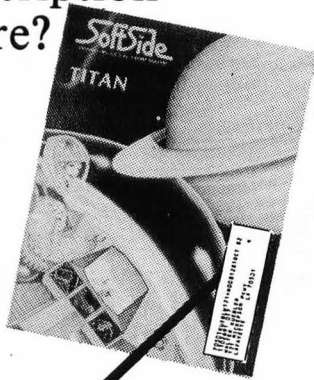
many private organizations attempting such an effort, some with a modicum of success, but the best of such organizations, given no governmental sanction, finds itself in a perpetual capitalization crisis. The field is in such a rapid state of flux that information can easily become outdated in a matter of hours, let alone months. In the past, the only effective method of establishing such a centralized information source and *guiding light* has been through government sanction or mandate. Educational television floundered in a state of disorganization until the Public Broadcasting System was formed. Only after several years under the auspices of a formally funded and governmentally endorsed organization was the system able to become more than a "talking head" droning at our children. Now, through such projects as the Children's Television Workshop, educational material is being presented which fully utilizes the possibilities of *instructional broadcasting*.

The time has come for an organization as extensive as the *PBS* (I speak of it as it was conceived, not as the struggling organization current budget cuts threaten to make it.) be formed to steer the production and application of computer software for the sake of our "national education." The US is in great danger of falling technologically behind the world. Indeed, in some fields we are already seriously lagging. This is not the time to be deemphasizing our former concentration on education. One of the main strengths which made the United States a continuing world power has been our concentration on internal capitalization. We must consider computer education, for students and the general public alike, another form of capitalization — an investment in our minds and those of our future generations.

Randal L. Kottwitz

Editor

When does your *SoftSide* subscription expire?



49007STANTG97*B00F1281OCT 82
1102691 017
GEOFFREY STANTON
97 BAYSIDE COURT
KALAMAZOO, MICH. 49007

The last five characters (three letters for month, two numbers for year) on the top line of your mailing label will tell you when your subscription ends.

For more information, write:
SoftSide
100 Pine Street, Holmes, PA 19043

See page 16 for ordering information.

EVERYONE'S TALKING ABOUT EdComTM'82

The unparalleled opportunity to expand your skills and knowledge at all levels and areas of interest in computer education.

**OCTOBER 21, 22, 23 and 24, 1982
LOS ANGELES CONVENTION CENTER**

EXPLORE the largest educational computer showcase ever offered!

SELECT from more than 350 hours of sessions presented by internationally known speakers from across the country who will address, evaluate, analyze, and share the phenomenal developments of computers in education.

GARY BITTER - *K-12 Curriculum*
 NICK BOHL - *Acquisition Funding*
 G. R. BOYNTON - *University Applications*
 HUGH BURNS - *Creative Writing*
 DAVID BYRUM - *Science Education*
 JUDY CHAMBERLAIN - *Strategies for the Gifted*
 SYLVIA CHARP - *Past and Future of Micros*
 JOHN CHILDS - *University Research and Management*
 BETTY COSTA - *Computerize Your Library*
 CHASE CRAWFORD - *Managerial Uses*
 SANDRA CUNNINGHAM - *Elementary School Literacy*
 DAN DAVIDSON - *Community College Applications*
 MARVIN EBBERT - *Telecourses in the University*
 JUDY EDWARDS-ALLEN - *Applying Evaluation Procedures*
 BOB ENENSTEIN - *Administrative and Classroom Management*
 WILLIAM ENGEL - *Instructional Computing*
 LEROY FINKEL - *Business Education*
 MARYLYN SUE FORD - *K-8 BASIC Programming*
 CAROLYN GILBREATH - *Support Services for Teachers*
 BOBBY GOODSON - *Districtwide Training for Literacy*
 RICHARD HARTNESS - *Hardware for the Blind*
 DAN ISAACSON - *Courseware Creation*
 MAX JERMAN - *Program Design*
 DEE LAMONT JOHNSON - *Special Education Software*
 ROBERT JUDD - *Simulations*
 KENT KEHRBERG - *MECC's Statewide Plan*

TOM KELLER - *Hardware Evaluation and Acquisition*
 RONALD LEMOS - *Acquisition of Resources*
 JOHN LOSSE - *Mathematics in Higher Education*
 ANDREW MOLNAR - *Is Education Keeping Pace?*
 DAVID MOURSUND - *Computer Literacy Without Computers*
 TED PERRY - *Authoring Systems*
 JIM POIROT - *Training the Teacher*
 M. D. ROBLER - *Reviewing the Reviewers*
 TERRI ROSEGRANT - *Reading Readiness/Special Education*
 BARBARA SADOWSKI - *Software Modification*
 ERNEST SAVAGE - *Vocational Curriculum*
 DAVID SHARPE - *Authoring Systems*
 GARY SHELLY - *Business Education*
 TWILA SLESNICK - *Policy and Curriculum Trends*
 VICKI SMITH - *Search for Software*
 RANDALL STICKROD - *Creative Uses of Graphics*
 KATHLEEN SWIGGER - *Preschool Software and Research*
 JOHN UNDERWOOD - *CAI Strategies*
 KATHY VOGT - *Pascal Workshop*
 WILLIAM WAGNER - *County Van Project*
 GEORGE WATSON - *Word Processing*
 J. FRED WEAVER - *Mathematics CAI*
 JERRY WILLIS - *Testing and Diagnosis/Special Education*
 JAMES WINEBRENER - *Educational Software Evaluation*
 KARL ZINN - *Applied Research in Higher Education*

AND MORE!

MicroCourses and Hands-On Sessions Will Feature

• TRS-80 • ATARI • PET • IBM • APPLE • TI

**Attend EdComTM'82!
EXPERIENCE THE EDUCATIONAL
EVENT OF THE YEAR!**

MAIL TO: Judco Computer Expos. Inc.
2629 North Scottsdale Road, Suite 201, Scottsdale, Arizona 85257

Name _____
 Address _____
 City _____
 State _____ Zip _____

Please send me registration information on EdComTM'82.
 Please include exhibitor information.

 **EdCom.'82**
SP



From our readers

INPUT

PLAYTESTING

Dear *SoftSide*,

A method of alleviating the cost of research and development has worked quite well for designers of wargames. I wonder if you have considered it for polishing and evaluating the many submissions you receive. While a wargame is still incomplete, the designers send it out to various clubs for "blind" playtesting. The clubs must respond in a set period of time. Changes are made, if appropriate, and a revised game is returned. In the end, the designers have a more polished product and the playtesters have a free copy of it. If the printed program was sent to the playtesters, they could develop enhancements to it. There would be some problems in the beginning until you obtain competent playtesters, but like the wargame industry, I think you would be satisfied in the end.

Dave Jameson
Boise, ID

Editor's Reply: *SoftSide* has utilized outside playtesting resources on a somewhat limited basis in the past. However, we are currently planning to expand our efforts in that direction with specific emphasis on computer clubs and users' groups. There are a number of details to be tended, such as nondisclosure agreements and procedures for verifying the quality of changes suggested by playtesters. We have a background sheet available. If your group is interested, write *SoftSide's* Editorial Department for more details.

COPY PROTECTION

Dear *SoftSide*,

It is our sincere belief that the microcomputer industry will benefit from the most complete exchange of ideas and information possible, consistent with the legitimate rights of software producers to adequate compensation. As authors and publishers, we recognize and understand the arguments for protection; but as users and enthusiasts, we oppose uncopyable and, by their nature, unlistable, unmodifiable programs.

One of the greatest pleasures of owning a personal computer comes from learning, and one of the best ways to learn is to review and try to understand what others have previously done. You can customize programs to suit your own specific needs or write your own with the knowledge gained. This is the way most of the early greats in micros learned. However, this is impossible when you can't list the program or make your own modifications — the learning process ends before it begins. Furthermore, a great many commercial programs would be

much more valuable to the user if modifications were possible (not to mention the much discussed problem of backing-up valuable, uncopyable disks). Locking the program in these instances actually makes the program less valuable to the user and may even reduce future sales. We believe this to be undesirable and that the risks involved in "going bare" are outweighed by the value to the user and the basic fairness of providing quality software which can be listed, understood, modified, and legitimately backed-up.

We are not, under any circumstances, condoning piracy! Duplicating another's copyrighted work and distributing it (even for free) without his permission is, and should be, illegal. Pirates are not romantic heroes; they are thieves who should be punished. However, we believe the vast majority of users are honest and will pay a fair price for good software. We are building our company on that belief ...and on faith in our customers.

Although almost all major software producers use some form of protection, there have been several recent announcements to the contrary. *Apple's* Mike Markkula stated recently that he would like to see the elimination of locked software altogether. Also, Mark Pelczarski of *Penguin Software* announced in March *Penguin's* decision to "go unprotected" with certain software. We applaud these efforts. It takes courage and trust in you, the user.

Thorne D. Harris III, President
Superior Software, Inc.

Editor's Note: The piracy debate goes on. *SoftSide* feels *Superior Software* has taken a rational approach to the situation and wanted to share their comments with you.

NEW ATARI® USERS' GROUP

Dear *SoftSide*,

I am pleased to announce the formation of an ATARI® Users' Group to be known as the Allentown/Bethlehem/Easton ATARI® Computer Enthusiasts, or in short — ABE'S ACEs. The purpose and direction of the group will be to further members' knowledge of the ATARI® computer and its programming by holding classes, demonstrations and discussion groups. Present plans are for the group to meet the first Saturday morning of each month.

For further information, we can be contacted at the address below.

ABE's ACEs
J.W. Mendola
Pres. Pro-Tem
Green Acres Park, Lot 2-8
Breinigsville, PA 18031

SoftSide

RADIO SHACK MODEM I

Dear *SoftSide*,

I must take strong exception to Lance Micklus' comments concerning the Radio Shack *Modem I*.

I'm not concerned with the quality of the *Modem I*. I have one, and it has worked perfectly. I use it for communicating with *The Source* (a local phone call, but a very weak carrier — much weaker than Lance's bulletin board, which is a long-distance call), and my mainframe at work (where I have maintained continuous contact for over twelve hours at a time without any problems). Please note that the local Radio Shack Computer Center must sell the *Modem I* by demonstrating it on a long-distance hook-up to *CompuServe*. If what Lance says was true, they wouldn't be selling any.

My main problem with Lance's comments is this: he has not done his homework. He has not used a *Modem I* himself, and his "exploits" in the Radio Shack Computer Center hardly qualify as scientific research. If people had listened to all of the bad press concerning the TRS-80® Model I when it first came out (the same type of comments that Lance repeats in his article to justify his position) nobody would own TRS-80® computers today. People who have problems with their equipment make noise. Those who don't have problems usually don't say anything. It's kind of a "silent majority" situation. I have owned my Model I for over two years. During that time I have had one failure — the RS-232 board went up the hill one day, causing the only unplanned reboot I've ever experienced. That problem was taken care of in a few days, and it was covered by the 90-day warranty.

Lance's trip to the Radio Shack Computer Center is a journalistic joke. He only proved that one particular *Modem I* was unable to get a carrier on his bulletin board at that particular time. It proved nothing else. Lance didn't try to connect to the bulletin board using another modem. This would have isolated the modem as the only variable in the system. But no, when he had the result that "justified" his thesis, he stopped his research. Maybe the phone line was bad; maybe the modem was bad; maybe it was sunspots. In any case, we'll never know. I certainly hope that Lance doesn't debug his software using this same rigorous technique.

All I ask is that you be more careful with your articles in the future. We don't need any more hearsay, illogic, and innuendo in our hobby than we have already.

Spencer R. Lepley
Tallahassee, FL

Lance's Reply: You are justified in feeling that my experience at the Radio Shack Computer Center in Paramus wasn't fair. I suppose I could get a *Modem I* and run controlled comparison tests here in Burlington to see if the results

changed. The problem is that I might get a *Modem I* that's really hot, giving me more favorable results than normal.

A far more valid test would be to have several thousand people use all of the reviewed modems under every imaginable condition. In a manner of speaking, this is exactly what I did, since the tests were based on the experience of thousands of *ST80* users. Their experience with *Modem I* was extremely poor when compared to their experience with other brands of modems.

I'm glad to hear that you are having good luck with your *Modem I* and have even been able to establish a connection on my own BBS. You'll be glad to hear that although I smoke more than two packs of cigarettes a day, I have not suffered from cancer, heart disease, or death. I still think the *Modem I* is a very poorly designed device and that smoking is bad for your health — in spite of the fact that your personal experience with the *Modem I* and my personal experience with cigarettes do not support either of these conclusions.

Editor's Note: For further comments from Lance concerning the *Modem I*, see *My Side of the Page*, later in this issue.

TRS-80® MODEL I JOYSTICK

Dear *SoftSide*,

Many, many thanks to you and Tigre Wenrich for your May, 1982, article, *Joystick Modification for the TRS-80®*. I have installed the ATARI® joystick on my TRS-80® Model I and it has really brought new life to many of my game programs. Long have I wanted a joystick, and now I can enjoy arcade-type games with authentic arcade style.

I would like to let any Model I owners know, however, that the installation is ever so slightly different than Mr. Wenrich described for the Model III. But first, let me assure anyone who is concerned about the "open-board surgery." I have absolutely no background in electronics and for years I have shuddered at the thought of opening my computer and dabbling around its innards. But, the thought of a joystick for an investment of only \$10 was enough to get me to try. I can say, from experience, that if you can hold a soldering iron steady, the job will be a simple affair.

The biggest problem I encountered is that the positive and ground solder points are not marked on the PC board. A little trial and error set me straight. To begin, remove the six screws and turn the keyboard upright again. Now, lift off the cover, lift the top PC board, and fold it forward. The underside of the keyboard is now in front of you and, if you can read letters upside down, you will see each set of solder tips is clearly marked. For each key, there are two solder points, one above the other on a slight diagonal. The positive is always the lower, left point. To this point, you should solder the corresponding wire from the joystick. If you have a numeric keypad, you will see that Radio Shack installed it in the exact same manner. One last point which was misleading in the article: the black wire cannot go to any negative post as Mr. Wenrich stated. I believe he meant any negative post of one of the keys to which you have connected another joystick wire. At first, I used the same post that Radio Shack used for the numeric keypad, and nothing worked.

If you want a more professional look, and do not want the joystick permanently attached to your computer, an additional investment of \$3 would help. I bought a male and female DIN plug, similar to the plugs already on the back of the unit. It has five pins and a ground, perfect for this application. I used Radio Shack Part Numbers 274-003 and 274-006 (to keep it in the family). I have the female socket sticking out the

expansion port opening about three inches and the male plug wired to the joystick. This added about twenty minutes to the job. Although I am getting no commission from Tandy for saying so, the \$3 is a small investment for the ability to detach the joystick whenever it is not needed.

So, if you read the article and drooled over the prospect of having a joystick, by all means, go ahead! If I can do it, anyone can. Thanks again to Tigre and your great magazine.

Robert M. Rosenfield
Warwick, RI

OUTPUT

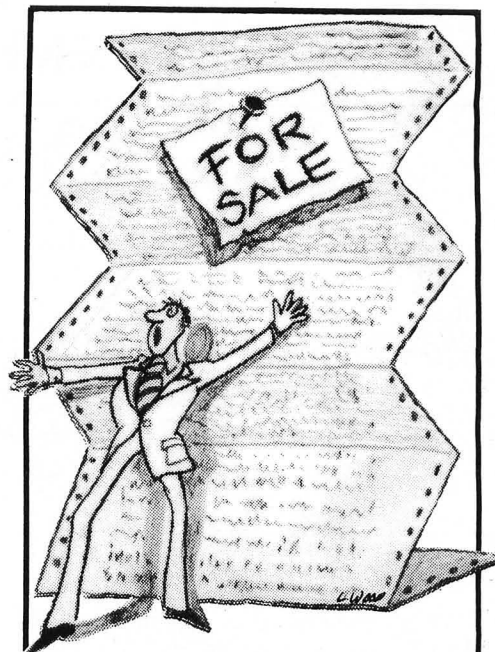
by Randal L. Kottwitz

In this and future issues of *SoftSide*, you'll find comments concerning the "inner workings" of *SoftSide*, formerly contained in *Outgoing Mail*, here in *Output*.

In any marketplace containing a large number of products, each attempting to carry a unique title, confusion is likely to abound. Such is the state of affairs in the entertainment software field. There are some blatant attempts at close emulation of titles in order to grab a portion of the same marketplace, but in most cases, the confusion is purely accidental. Such seems to be the case with a product reviewed by Bruce Chapman in the May *SoftSide*. Bruce gave rather unfavorable comments to an ATARI® program entitled *Forest Fire* (part of a package called *Out-door Games*). Shortly after the issue was mailed, *SoftSide* received a letter and disk from Arthur M. Walsh of Artworx Software, stating that Artworx publishes a program, *Forest Fire*, for the ATARI® which does not have the faults we mentioned in our review. We checked the disk, and although the two programs certainly are similar in concept, the Artworx *Forest Fire* is much better implemented. Mr. Walsh also informed us that Artworx is about to issue an enhanced version of the game to be titled *Forest Fire 2*. One must wonder what the plans are of the publisher of the other *Forest Fire*. The impact of this sort of confusion on a single title is minor — it does, however, point up something of which software authors should be aware. Name your program as carefully as you would a book. It may show up in a computer store window in your neighborhood.

We've had a bit of confusion over the proper address for correspondence with *SoftSide*. You'll note several changes in the "masthead" on the table of contents page this month. One is that all editorial and advertising correspondence should be directed to our offices in New Hampshire. This should also include any specific questions concerning articles, programs and computing. The address listed in Pennsylvania is for the subscription fulfillment house which handles all of our subscriptions, product orders, back issues, and customer service. Sorry to say, they won't be able to tell you how much memory a program requires or what DOS is required to run a *SoftSide* disk.

That's all for this month. Enjoy issue #31, it should give your gray matter a good workout. Until next month — Happy Hacking! ☺



SELL YOUR PROGRAM AND KEEP IT TOO!

One of the nicest things about selling your program to *SoftSide* is that it's still your program after we buy it. Actually, what we are buying is the right to publish your program once in our magazine and on subscription disk and tape. This is what we call "one-time rights." Three months after your program appears in *SoftSide*, you are free to sell it again to anyone. And, now that it's been published, your program is worth more. So send today for a copy of our free author's guide and find out how you can sell your program and keep it too.

Write to:

SoftSide Publications
Dept AG1
6 South Street
Milford, NH 03055

HINTS & ENHANCEMENTS



From our readers

ATARI® Cross Reference

Here's an improvement for the excellent program in your recent *SoftSide*, May 1982, Pg. 62 — ATARI® *Cross Reference*. The program cannot process a statement such as:

```
10 PS = ""
```

Such statements generate an Error 5 at Line 3050. I have found that the following changes correct the problem:

```
1008 IF ASC(REG$(LAST))=34 THEN
GOSUB 3050
3020 IF ASC(POS$(H,H))=34 THEN 3050
```

I hope this proves useful to *SoftSide's* readers.

Lee Hopkins
Sacramento, CA

Printout Using The ATARI® 850 Interface

Our thanks to reader Jeffrey Kerner of Woodlands, TX, for the following information. It pertains to ATARI® programs which use the LPRINT statement, such as *Microtext* and *Banner Machine*.

When using the 850 Interface with a printer, an LPRINT line which ends with a semicolon apparently causes the ATARI® to pad the line with as many as 40 spaces. An alternate approach that doesn't have this problem is to open an IOCB for the printer and then print to that IOCB number. The lines listed below are Mr. Kerner's modifications to *Microtext*, which illustrate the technique.

```
5 GRAPHICS 0:POKE 764,255:POKE 709,4:P
OKE 710,15:REM TO REDUCE EYE STRAIN
20 OPEN #7,8,0,"P:"
200 POSITION 2,0:?"^SAVE ^LOAD ^REVIE
W ^EDIT ^PRINT ^QUIT"
7070 ? CL$:? #7;"";P$="":CR=0:I=0
7130 ? #7;S$(1,LH);L$;
7150 ? #7;""
7590 ? #7;S$(1,LH);PP$;
```

```
7610 FOR J=1 TO LS:LIN=LIN+1:?"#7;"";N
EXT J
7615 IF LIN>59 THEN FOR J=1 TO 66-LIN:
?"#7;"";NEXT J:LIN=0
```

TRS-80® LADDERS

Rik Pierce, the author of *LADDERS* (*SoftSide*, May, 1982) has sent us a substantial enhancement to his program which allows one or more human players to play against one or more computer opponents.

Variables

HP: Number of human players
NP: Current player number
S(1-2): Value of pairs of dice
DT(1-4): Value of each die
DS(1-6): Values of all possible pairs
V and V1: Value of chosen dice
FM: Flag for marker checks possible move
M: Number of markers on board
V(2-12): Available ladders
 0 = available 1 = your mark
 2 = unavailable
R1(2-12): Number of rungs on each ladder
L (NP , MX(X) , U)
 (player , ladder , rung)
This keeps track of who is where
P(M(X),Y): Board location to print marker
Z(X): Computer choice
SS: Holds sound information for
K = USR(0)

```
300 PD=121:PO=896:PB=30:PL=282:PW=PL+388
320 'VARIOUSTEMP
325 NT$(1)="HEAD ON":NT$(2)="DEVIUS":NT
$(3)="OFF THE WALL":NT$(4)="SCATTER BRA
IN"
900 S$=S2$:PRINT@0,CS:PRINT@384,"HOW MAN
Y HUMAN PLAYERS?":B=USR(0)
920 GOSUB3440:HP=VAL(A$):IFHP<10RHP>4THE
N920
955 FORX=1TOHP
960 IFS$=S1$THENS$=S2$ELSE$=S1$
965 PRINT@384,CS;"NAME OF PLAYER"X;:B=US
```

```
R(0):INPUT N$(X)
970 NEXT
975 PRINT@384,"HOW MANY COMPUTER OPPONEN
TS? 0 TO ";4-HP
980 GOSUB3440:TP=VAL(A$)+HP:IFTP>4THEN98
0
981 IFTP=HPTHEN1020
985 PRINT@384,CS"CHOOSE YOUR OPPONENTS:
1 ..... "NT$(1):PRINT"2 ..... "NT$(
2):PRINT"3 ..... "NT$(3):PRINT"4 ..
..... "NT$(4)
990 FORX=HP+1TODP
995 GOSUB3440:P=VAL(A$):IFP<10RP>4THEN99
5
1000 PRINTNT$(P)
1005 N$(X)=NT$(P):FL(X)=X
1010 NEXTX
1015 FORTIM=1TO1000:NEXT
1390 IFN$(NP)=NT$(1)THENZ(1)=2:Z(2)=12:Z
(3)=8:Z(4)=6:Z(5)=7
1392 IFN$(NP)=NT$(2)THENZ(1)=8:Z(2)=6:Z(
3)=7:Z(4)=2:Z(5)=12
1394 IFN$(NP)=NT$(3)THENZ(1)=3:Z(2)=9:Z(
3)=10:Z(4)=4:Z(5)=5
1396 IFN$(NP)=NT$(4)THENZ(1)=RND(4)+1:Z(
2)=RND(4)+RND(4):Z(3)=RND(3)+RND(3)+RND(
4):Z(4)=12-RND(4):Z(5)=RND(11)+1
1850 IFFL(NP)THEN2800
2450 IFFL(NP)THEN2920
2680 IFL(NP,M(X),R1(M(X)))=NPTHEN W(NP)=
W(NP)+1:LD=LD+M(X):IFM(X)>7THEN D=LD-(M
(X)-7)*2 'SKEWS COMPUTER PLAY WHEN TOP
OF LADDER REACHED
2800 'COMPUTER PLAYS
2810 V=0:V1=0
2811 IFLD<15THEN2820
2820 FORZ=1TO5
2830 FORX=1TOD3:FORY=X+1TOD4
2840 IFDT(X)+DT(Y)=Z(Z)THENIFV(Z(Z))<2TH
ENV=X:V1=Y
2850 NEXTZ,V,X
2855 IFVTHEN2890
2860 V=RND(4)
```

```

2870 V1=RND(4):IFV1=VTHEN2870
2880 IFV(DT(V)+DT(V1))>1THEN2860
2890 PRINT@PD,CS"
                "N$(NP)" CHOOSES ---
-----> ";
2900 PRINTCHR$(27)D$(DT(V));" AND ";:FOR
TIM=1TO700:NEXT
2910 PRINTCHR$(27);:GOTO1980
2920 IFM<3THENP1=100-LD:GOTO2985
2930 P1=0:N=0:FORX=1TO3:N=M(X)
2940 IFV(N)=2THEN2970
2950 IFM(X)>7THENN=M(X)-(M(X)-7)*2
2960 P1=P1+N
2970 NEXTX
2980 P1=(P1-5)*7-LD*2-TC*3
2985 PRINT@PD,CS,:;:IFFL(NP)=2THENIFP1<60
THEN2998ELSE2995
2990 IFRND(90)>P1THEN2998
2995 TC=TC+1:PRINT@PD,CS;N$(NP)" CONTIN
UES";:GOTO1600
2998 TC=0:PRINT@PD,CS;N$(NP)" QUILTS HERE
";:GOTO2520
2999 STOP
3000 PRINT@PD,CS"      W I N N E R      * *
*      "N$(NP)"      * * * W I N N E R"

```

Apple™ Format Subroutine

Here's a very short subroutine for the Apple™ computer which converts any number to a proper "dollar" format. The way the Apple™ is designed, any unnecessary zeros to the right of the decimal point are omitted. However, when working with reports, budgets, or anything where a dollar format is required, it is often desirable to have figures printed out in a consistent format with two decimal places. The following subroutine will make the necessary conversion.

To use the routine, simply set the variable Y1 equal to the number you wish to adjust, then go to the subroutine. The adjusted number is printed as the variable Y1\$. Here is an example:

```

80 REM SUBROUTINE TO PUT NUMBERS
   INTO $ 0.00 FORMAT
81 Y1 = INT (Y1 * 100 + .5) / 10
   0:Y1$ = STR$(Y1):L1 = LEN
   (Y1$): IF L1 = 1 THEN GOTO
   85
82 IF MID$(Y1$,L1 - 1,1) = "."
   THEN Y1$ = Y1$ + "0": GOTO
   86
83 IF L1 < 3 THEN GOTO 85
84 IF MID$(Y1$,L1 - 2,1) = "."
   THEN GOTO 86
85 Y1$ = Y1$ + ".00"
86 RETURN

```

I hope this can be useful to other readers.

David Marshall
Champaign, IL ☺

How would you like

A
Free

DISK OR CASSETTE SUBSCRIPTION?

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

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

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

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

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

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

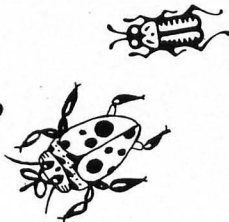
- Your entry should incorporate any improvements and enhancements that you can add to make it an even better program. Don't feel that you have to limit yourself to the boundaries of the original. (On the other hand, don't go overboard and destroy the character of the original by completely rewriting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

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

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

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

Bugs, Worms, and other Undesirables



There is a bug in all three versions of *Solitaire*. May, 1982 which keeps the first card in the deck from being shuffled with the rest. Change line 1230 to read as shown to correct this.

Apple™ Version

```
1230 FOR I = 51 TO 0 STEP -1: X
    = INT ( RND (1) * (I + 1))
    : T = D(X): D(X) = D(I): D(I) =
    T: NEXT
```

ATARI® Version

```
1230 FOR I=51 TO 0 STEP -1: X=INT(RN
D(0)*(I+1)): T=D(X): D(X)=D(I): D(I)=T
: NEXT I
```

TRS-80® Version

```
1230 FOR I=51 TO 0 STEP -1: X=INT(RND(0)*
(I+1)): T=D(X): D(X)=D(I): D(I)=T: NEXT
```

In the Apple translation of *Micro-Man* (May, 1982), running over an energizer (\$) does not add the point value of the energizer to the player's score. This can be corrected by adding the following line.

```
607 IF PK = 100 THEN SCR =
SCR + 50
```

CALENDAR

August 1-4 Microcomputer Applications In Education Workshop Cloud's Cal-Neva, Lake Tahoe, NV

The University of Nevada — Reno Division of Continuing Education and the Washoe County School District will sponsor this workshop for school teachers and administrators.
Contact: Shirley Beck, Division of Continuing Education, Reno, NV 89557.

August 21 The Third Annual Apple™ Fair New York University, New York, NY

The Big Apple Users Group of New York will sponsor this fair, which will stress both business and leisure applications of Apple™ hardware and software.
The program will include general business application classes and lectures in pre-packaged software, as well as lectures and "hands-on" activities in the realm of graphics, games, and education.
Admission is free.
Contact: Barbara McMullen, (914) 245-2734.

August 12-13 Microcomputers In Vocational Education Conference Sheraton Inn, Madison, WI

This conference, for all persons interested in microcomputer applications to vocational education programs, will provide access to both computer information for beginners and to more advanced applications of software programs for vocational education curricula. A handbook, distributed to conference participants, may also be purchased separately (\$20) by those unable to attend.
Contact: Judy Rodenstein or Roger Lambert, Vocational Studies Center, 964 Educational Sciences Building, 1025 West Johnson Street, Madison, WI 53706.

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, at least three months in advance, 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.

BUY! SELL! TRADE!
COMPUTER & HAM EQUIPMENT
COMPUTER®
TRADER

**PERMANENT
SUBSCRIPTION
\$10.00**

Low Ad Rates — Mailed Monthly

COMPUTER TRADER®

Chet Lambert, W4WDR
1704 Sam Drive • Birmingham, AL 35235
(205) 854-0271

Please include your Name, Address, Call Sign or Phone Number



Motion Graphics For The 21st Century

by Fred D'Ignazio and Allen L. Wold

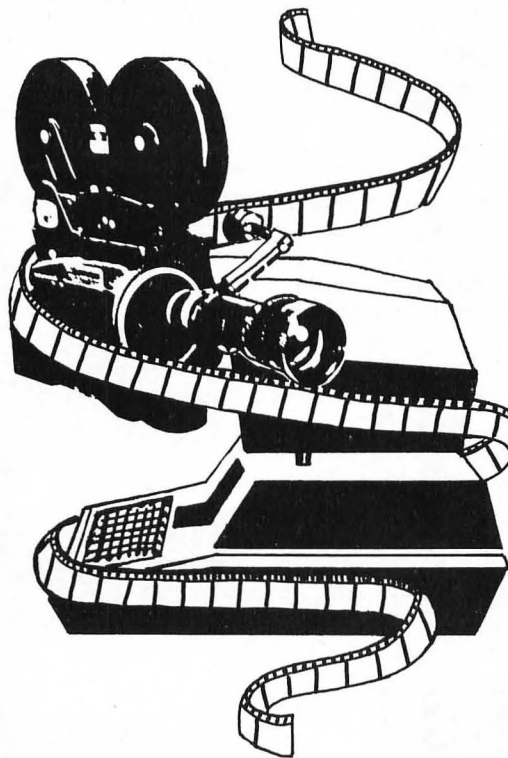
John Whitney, Jr. is an expert at making computer movies that are colorful and realistic. He is a computer pioneer with startling ideas about computers of the future. For the last fifteen years, Whitney and his father, independently, have been using computers to create special effects for TV programs, commercials, and popular movies such as *Star Wars*, *2001: A Space Odyssey*, and *Superman*.

Whitney recently started his own film company, with the help of his partner of many years, Gary Demos. Demos is a computer hardware wizard, and he and Whitney are building a machine so powerful they call it a "nuclear power plant" computer. When it is finished, it will generate pictures that are so realistic, they will be indistinguishable from a movie's live photographic portions. Whitney feels that computer generated movies will be one of the main sources of entertainment in the future. These movies will go far beyond the special effects that today's computers create.

Most computer special effects today are used in TV commercials, cartoons, and science fiction/horror films. Whitney and Demos want to create computer effects for use in TV documentaries — tours through outer space and the microscopic world of the atom. They think their computer will be able to create actors and scenes for realistic and fantasy films alike. A computer that can draw people, atoms, and planets that look real must become more artistic than those in use today. Whitney plans to build a computer modeled after the talents of the great artist, Leonardo da Vinci.

Leonardo da Vinci had a gigantic thirst to learn how the world works. For example, instead of reading other people's accounts of the way organs functioned inside the human body, da Vinci cut up cadavers and made draw-

ings of what he saw. During his life, as he learned more about the world, da Vinci's drawings became more lifelike and dynamic. For example, his drawings of birds and imaginary airplanes are based on his study of bird wings during long hours watching birds in flight. Da Vinci could draw the world in motion because he understood how the world worked.



John Whitney wants to build a computer that works on the same principle. First, Whitney will create a huge "experimental database" for the computer to access — an automated library of the real world. Then, when Whitney asks his computer to create a scene of violent weather, it won't just recall a "canned" cloud cartoon and make it move. Instead, the computer will

search its memory for characteristics of real weather — its physics, dynamics, chemistry, and optical characteristics. Based on that knowledge, the computer will create mathematical models of dark, rumbling thunderheads, a jagged plexus of flashing lightning, and gusting sheets of wind-driven rain. It will then translate these models into three-dimensional images moving across the screen.

Whitney feels that his "da Vinci" computer will revolutionize the motion picture business in the 1980's and 90's. He and Demos, he says, are working on "motion graphics for the 21st century."

The Computer As Creator

Whitney has other ideas that are even more extraordinary. He won't be satisfied with an artistic computer that can create simulated creatures and events. He wants his computer to learn how to create *real* creatures and events, and then breathe life into them. He wants the computer's swirling thunderstorm, its ferocious dinosaur, or its medieval French knight to act as if they are truly alive. Once they possess that *life* they will no longer be mere images on a movie screen. Instead, they will act unpredictably, just as they would in real life.

Just think of a movie filled with dozens or hundreds of creatures, scenes, and events that are dynamic, fresh, and unpredictable. This movie would be more frightening and enchanting than any we've ever seen. It would capture us and carry us away to the world inside the movie, a world in which these creations really lived. Someday, Whitney plans to create "Disneyworld" electronic amusement parks, inhabited by intelligent, lifelike aliens, monsters, human heroes and villains — all created by a computer.

He foresees a computer fantasy-game "Olympics" broadcast via satellite to a world-wide audience. People will watch on their computerized TVs as electronic "gladiators" from each nation put on wired helmets and are transported into a dangerous world of fantasy, invented by the computer.

A New Form of Life

We are witnessing the dawn of a new era, the birth of a new species of life, one which will be strange and exotic. It won't have flesh, blood or bones. Its body will be made from silicon crystals, plastic, and metal. In a primitive state, this new life form already exists. It is the computer.

But how could that be? Compared to human beings, today's computers are like cave-dwelling savages. They have only limited intelligence, and their senses are rudimentary — on the level of an earwig or a turtle. But, we need to recall that human beings, in a time-relative sense, have a huge head start on computers. Humans have been evolving for billions of years. Computers have been around for less than forty. Also, humans are evolving at the pace of a snail. In comparison, computers are progressing at the speed of a jack rabbit.

The first computers were giant "dinosaurs," nurtured and sheltered in scientists' labs. They were heavy, massive, and dim-witted — good at arithmetic, but miserable at almost everything else. Today, millions of computers are appearing in thousands of forms, all over the world. They are faster than their "ancestors," far more rugged, and far more intelligent. They see. They talk. Some are even learning how to walk. They are taking their first, halting steps, just like a human baby. Perhaps that's the way to view computers — as babies. And we, the human beings, are their parents. It is our job to teach them about themselves and the world, and to help them grow.

John Whitney is just one of a large number of respected computer scientists who share this view. In the opinion of many, by creating the computer, we have not just invented another gadget or machine. Instead, we have given birth to a new form of life. The real issue facing society, then, is not *artificial intelligence*, it is *artificial life*.

According to Whitney:

"Whether we like it or not, mankind's child is the computer. This is an inevitable process that can no longer be stopped. Maybe this is the purpose of all biological life: to give rise to the next form of life."

Digital Moviemaking

What does *artificial life* have to do with movies? More than you might think. Movies are undergoing rapid and fundamental changes. In a few, short years they will become a form of entertainment totally unlike what we know today.

From the beginning, movies have been created from film — individual photographs chained together on a strip of celluloid, flying by on a picture screen to give the illusion of movement and life. Now, filmmakers are experimenting with *digital movies* — motion pictures created and stored electronically on a computer. Film is either unnecessary or created only in the final step, to allow projection on standardized theatre equipment.

Famed movie director, Francis Ford Coppola, of Zoetrope Studios, is working with Japanese and American computer and video manufacturers to develop a *director's digital workstation*. According to Coppola, a director will be able to sit at his computer-based workstation and create complete sections of a new movie, simultaneously displaying and manipulating as many as 1000 small video images. The director will operate various buttons, dials, light pens, and touch-sensitive screens to create, edit, cut and paste together live scenes, animation, special effects,

and a digitized soundtrack, all called up from the computer's memory storage.

Coppola's workstation is just the beginning. Once filmmaking is digitized and transferred to computers, directors will no longer spend most of their time on Hollywood sets or on location around the world. Instead, they will sit in front of a computer, manipulating trillions of bits of visual and aural data to shape a new movie.

Digital filmmaking will give directors new freedom. As Coppola says, "The display is your imagination." Physical limitations — sets, props, actors, budgets — will cease to be major constraints on a director's creativity. The ideal film could be within his grasp.

Interactive Multi-Movies

Movies may soon be produced differently, but, will they look any different to the average movie-goer? What will the movie theatres of the future be like?

Imagine entering a movie theatre ten years from now: You receive a pair of small, plastic goggles, to make the movie appear to be in 3-D, much like the 3-D movies of the 1950's, but of much higher quality. The computer transforms a digitized movie from two to three dimensions at almost no extra cost.

You choose Theatre A, where the new *Intelligent Movies (IMs)* are shown. The lights dim. Your chair becomes transparent. All four walls, the ceiling, and the floor become a single, enormous screen. No projectionist or projector is required. The flat, scuff-proof panels underneath and around you display colorful, high-resolution pictures without them. They are under the control of a powerful minicomputer located in the theatre's video control room. The minicomputer produces the displays at high speed in rotating, 360 degree arcs.

The movie begins. But you aren't watching the movie. You are in it! With a shock, you jerk back into your padded seat. You are at the Grand Canyon, perched on its ragged lip. The mouth of the canyon gapes fantastically wide before and beneath you. It threatens to swallow you like a speck of dust. You look up. The sky overhead is a deep blue. You feel suspended in mid-air, infinitely small. You can see dozens of miles in all directions. A thunderstorm approaches, and down below, you see a small party of backpackers hastening up the trail on the side of the canyon, trying to sprint ahead of the rain.

MOVING?

If you're planning to move, please let us know at least six weeks in advance. This will help us to change your address insuring you with prompt and accurate service on your subscription. Attach your current mailing label filling in your name and NEW address in the space provided.

Name _____
New Address _____
City _____ State _____ Zip _____

Send old label with your
name and NEW address to:
SoftSide
100 Pine Street
Holmes, PA 19043

You saw this same movie last night, but it was completely different. It was raining when the movie started. You were on the canyon floor. High overhead, a bolt of lightning lanced out of the sky and tore off a section of the cliff. Boulders the size of houses clattered and crashed around you.

You relax, sit back in your seat, and smile. No wonder it's different — this is an *IM*, an *Intelligent Movie*. You could see it every night for a month, and each night it would be different. The various scenes and plots would change, as would the characters. Most importantly, the action would be unique every time you saw it. The differences would be totally unpredictable.

Huge numbers of plot parameters are under control of the computer. Each night, the computer varies those parameters randomly within certain limits. Those limits keep the plot realistic and dramatic. Otherwise, the computer's changes might be choppy, discontinuous or distasteful. Embedded in the computer's complex, ROM-based software are the key elements of the moviemaker's craft, mined from the most brilliant directors in the world.

All three theatres are showing *multi-movies*. But, Theatre B is showing multiple movies simultaneously, a *Viewer-Controlled Movie (VCM)*. You are seated in a soft, padded seat. The armrests are miniature control panels, studded with buttons, dials, and small screens. You pick up a "video head-*phone*" on the corner of the armrest and fit it over your eyes and ears. Now you are wired to the movie, and it is wired to you.

The movie begins the same for everyone. Soon, however, twenty or thirty different movies are running simultaneously. You and the other people in the audience are creating the movies, on the fly, and watching them on the miniature, stereoscopic, 3-D video screens wrapped like doughnuts around your heads.

When a movie "plot branch" occurs, you are notified by a blinking red light in the corner of your peripheral vision. You can flick a switch and choose which direction you want the plot to go (a menu flashes before you), or you can do nothing, and the movie takes a default branch and continues with no interruptions. At any time during the movie, you can switch to anyone else's personal movie in the theatre. You don't know whose movie it is, but, you can casually peek into the other people's movies, browse and loiter, watch the whole thing, or zip back to your own.

You can also freeze your movie, run it backwards, or jump to the end, the beginning, or the middle. After it is all over, you can play the role of a movie critic. You can "can" it, if it was miserable, or give it high marks if it was good. The computer stores your movie and assigns you a private "Movie ID Number" so you can return and watch your personal movie again on another night. The movie will remain stored for a full year in the computer's private movie library and be available to you on recall or for purchase or lease for your own home video system.

Theatre C is showing a *Consensus Movie (CM)*. As in Theatre A, the audience only watches one movie at a time. But this time, instead of one screen, there is a large screen on the far wall, complemented by several smaller screens, scattered throughout the room. As in Theatre B, you plug yourself in and your seat is surrounded with controls. But, this time you don't choose the plot on your own, you do it with the majority of your fellow viewers — by *consensus*. Votes are taken at the beginning and at key points during the movie. Whatever the majority of the viewers would like to see is what is shown.

However, you can still exercise individual creativity. Depending on what

seat you chose, you get to control one of the small, border screens surrounding the main screen or located elsewhere in the room. As a "scene editor," you can choose from a hundred different perspectives — zoom, close-up, panoramic, character shot, text, abstract, etc. — to flash on the smaller screen. The majority of the audience has chosen the main feature and its branches, but you can embroider its edges with exciting images.

You might see the movie in any theatre — A, B, or C — several nights in a row. You would see a different movie each night, governed by your choices and those of your fellow viewers.

The Technology Beneath The Magic

We have explored some of the exciting possibilities for movies in the near future. So far, we've looked only at the magic and not the technology which supports it. In the next two episodes of *Entertainment Tomorrow*, we will examine the technology that will be used in future movies. We will see some of the new developments in computer graphics and computer animation that will make it possible to realize movies and movie theatres like those we've just envisioned. ☺



"YOU'RE FOUNDER & PRESIDENT OF THE THIRD LARGEST SOFTWARE FIRM IN THE COUNTRY?"



My Side of the Page

By Lance Micklus

It looks like some *SoftSide* readers with Radio Shack Modem I's did not agree with my evaluation in April. Several of you called my own MOUSE-NET™ system to check it out with your own Modem I's and to say that everything worked perfectly. One caller, Pat Tancreti, was using *ST80-III*, so MOUSE-NET™ was able to test the connection automatically and include the results in the user log. Since the other callers were using older *ST80™*'s or some other terminal program, MOUSE-NET™ could not do the automatic test for telephone reliability. It entered their telephone line reliability as 0 even though it obviously could not be so. The messages they left follow with their location after their name in parentheses:

To: Lance Micklus
From: Ric Krasch (Rialto, CA)
Date: 4/10
Time: 10:52

"Lance, as ever, your article in SoftSide is fantastic. However, I, being curious, am trying out my new Modem I. It seems to get thru OK, even from California! And twice! Just thought I would let you know the results....Everything here is crystal clear and perfect. You must have gotten a lemon that one time in Paramus. Keep the good info coming, as I look forward to reading it every month."

To: Lance Micklus
From: Spencer Lepley (Tallahassee, FL)
Date: 4/7
Time: 20:05

"Just a note in response to the April SoftSide — not everyone has problems with Modem I. My TRS-80® Model I and Modem I with Modem80 software is having no trouble working long distance to your bulletin board. Maybe there is unit-to-unit variability or maybe some Modem I's need to be tweaked up a little."

To: Lance Micklus
From: Steve Reisenauer (Racine, WI)
Date: 4/6
Time: 15:09

*"Hi Lance,
I read your article on Modems in SoftSide. What you said about Modem I users having problems has not happened to me at all. I have been using the Model I for almost a year now, have dialed across many miles, and have yet to discover any problems. As a matter of fact, my use of your BBS right now is proof!*

By the way, I am using your ST80™ program as well. I just want to say it's a great program and I've had great success with it. However, I will be buying the terminal package Omniterm in about a month."

To: Lance Micklus
From: Pat Tancreti (West Haven, CT)
Date: 4/4
Time: 01:21

"Lance: I just finished reading your article in the April SoftSide about the Modem I and wanted to see if I could get through. I did!"

To: Lance Micklus
From: Gary Lamber (Magna, UT)
Date: 4/3
Time: 18:14

"SoftSide about Modem I — I am calling from one and have had no trouble with it when calling a BBS or Data Base System."

UPS

I ran across an interesting article recently about the boom in the UPS business. No, I don't mean the brown trucks full of packages, but the companies that make Uninterrupted Power Supplies (UPS). Many computer centers, getting more concerned about power outages and the damage they can do, are investing in UPS systems.

For example, one computer center said they use their computer at night to invest money. It's not unusual for them to loan as much as fifty million dollars on a short-term loan for one day. The interest on such a loan might be as high as \$50,000. If the computer goes down and they can't make the transaction, that money is lost. Eliminating this risk can justify a UPS system in a hurry.

Small computer users, too, are starting to find more and more need for UPS systems. Certainly, if you're running a computer bulletin board, a power failure can raise the devil. Also, more and more people are going to hard disk. When you blow up one of those, you really have problems. It takes me 35 minutes just to format my 10 megabyte hard disk, and I don't look forward to restoring all the files on it twice a month due to power failures.

It seems to me that a company like Radio Shack ought to be able to design a UPS rather easily. If they weren't priced too high, I'll bet they'd sell a ton of them.

The Whereabouts of Lance Micklus

These days, I definitely do not feel like a computer programmer. I've been spending most of my time writing new *ST80™* documentation. As I'm writing this, it still isn't finished, but it looks like it will be running about 100 pages (25,000 words). It certainly points up that there's a lot more to programming than just writing the program.

In the middle of that project came an annual commitment to my old employer — Vermont Educational Television — to help them with their on-the-air auction.

During the auction, I had a chance to catch up on some

reading. One magazine was devoted to communications — television, cable, phone networks, satellites, and the like. Its subject was that famous line, “In a few years, everybody will have...” Specifically, it was referring to satellite communications. Even the poorest third world countries are now receiving service from the “birds in the sky.” Rural villages in Africa, where the natives still wear grass skirts, have a satellite.

There’s one problem. It’s called “the last mile.” Yes, these primitive natives have their own satellite, but, they don’t have telephones to make the calls using it. They don’t have televisions to receive programs from it. The fact of the matter is, a lot of them are starving to death. They really don’t care what the satellite can do for them and would gladly trade their “bird” for a bowl of rice.

What many of the people who make statements of future profundity and bliss forget is that we live in a rich Western nation where people buy computers for toys. Even if you gave computers and television sets away for free, there would be many people on the face of the Earth who wouldn’t want them because they’d be of no use.

Let’s use the telephone as an example: It would cost the third world nations \$5 trillion dollars to install telephone equipment that would provide the same kind of service enjoyed by the rich Western nations today. There isn’t any way third world nations can afford to spend that amount of money. If they had it, telephone service wouldn’t be their first priority.

The new technology from the West is spreading the gap between the haves and the have nots. That’s the stuff that wars are made of. You certainly can’t expect IBM or Radio Shack to turn their production plants into rice fields to feed the world. Yet, in a funny sort of a way, it is their problem, just as it is yours and mine.

Next time you hear somebody say that in five years everybody is going to own a super wiz-bang, don’t take it too seriously. There’s more to this world than super-wiz bangs, like food.

The Mean Checkers Machine

A lot of people have been getting upset because of *The Mean Checkers Machine*TM and the problem with wide characters on the Model III. I don’t blame them. I’m upset too. I’ve reported the problem to anyone at Radio Shack who would listen to me since the day I got my Model III. To date, nothing has been done about it.

Part of the problem is that there is another, totally unrelated, problem in the ROM in 32 character mode. Everybody seems to set it aside as “that problem.” The ROM has nothing to do with it. What’s wrong is that when a Machine Language program writes into the video memory in 32 character mode, the CPU appears to be skipping one instruction. It does this only once in a while and it appears to have something to do with the clock interrupt. The problem can be severe in some machines, non-existent in others.

Somebody on MicroNET found a way around the problem. Place NOP instructions after every instruction modifying the video memory. That way, if the CPU does skip an instruction, it’s harmless. That trick is now used in version 2.1 of *The Mean Checkers Machine*TM and it works very well. Unfortunately, that doesn’t help those of you with version 2.0 (the version published on *SoftSide DV*, November, 1981) who are trying to use it on the Model III. I can tell you, however, that there is no easy patch. The fix requires putting one-byte instructions in the program in several different places. It sounds simple, but it’s not.

The only thing that can be done is to replace your version 2.0 with version 2.1 if you own a Model III. Unfortunately, that’s not simple either. It costs money to handle an order. (I can sell you a single paper clip for \$5.00 or a box of 100 for \$5.89.) I’ll agree that it isn’t fair for you to pay for a problem that isn’t your fault. On the other hand, it’s not my fault, it’s not *SoftSide*’s fault, and you can’t blame Radio Shack either. After all, Radio Shack didn’t sell you the program and they never said it would work.

All that can be done is to upgrade your version 2.0 to version 2.1 for \$7.50. The folks at *SoftSide* have agreed to handle the orders. Just send your original tape or disk of *The Mean Checkers Machine* or November, 1981 *SoftSide DV* with a check or your charge card number (VISA or MasterCard) and we’ll fix it up for you. Don’t forget to include a note explaining what you want done. Here’s the address:

SoftSide Publications
Department MCM
6 South Street
Milford, NH 03055

If you feel this is unfair and it will make you feel better, you can also include a letter complaining about how you feel ripped-off. I won’t argue with you and I won’t blame you. After all, the only difference between version 2.0 and version 2.1 is a couple of NOP instructions. I do feel somewhat responsible, but can find no other answer at this time.

Why Do People Get Ripped-Off?

Since we’re on the subject and, lately, everybody seems to

VERVAN Software
Utility Programs for the ATARI 400/800*

CASDUP
A machine language program that allows you to copy ANY Atari cassette. Cassette Only **\$20.00**

CASDIS
A machine language program that converts a cassette boot program into a disk boot program. Disk Only **\$25.00**

DISDUP
A machine language program for duplicating hard-to-copy disks. This is more than a simple copy routine. Disk Only **\$25.00**

VARMAP (Available in September 1982)
A machine language program that lists the variables in a BASIC program along with the number of every line that references them. Specify Cassette or Disk **\$25.00**

LINMAP (Available in September 1982)
A machine language program that lists each line in a BASIC program along with the number of every line that references them. Specify Cassette or Disk **\$25.00**

FULMAP (Available in October 1982)
A single program that includes all the features of VARMAP and LINMAP. An additional feature is that variables used in indirect addressing are listed separately. Specify Cassette or Disk **\$40.00**

Please add \$2.00 shipping and handling per program. California residents please add 6% state sales tax.
Send check or money order to:

VERVAN Software
10072 Balsa Street
Cucamonga, Ca. 91730

Dealer Inquiries Welcome

*ATARI 400/800 is a trademark of Warner Communications, Inc.

be talking about how badly computer consumers are treated, it might be a good time to talk about why people get ripped-off. About a year ago, when Lance Micklus, Inc. got its new photocopier, I got a call from a company in California that sells toner. They claimed there was going to be an increase in the price of our Canon® toner. Since they couldn't get their sales literature to us in time, we could order by phone if we wanted to. The new price was going to be \$145.00, but if I ordered now, they would sell it to me for \$108.00 per box. I decided to order a box, but was shipped two boxes instead of one. With shipping charges, the cost totalled \$245.00.

Fool that I sometimes am, I paid the bill promptly. When we got around to using the toner, I suddenly learned why this stuff was sometimes called "California toner." It was awful. The copies it made were muddy. The waste developer tray, which normally fills only once a year, was full in a week. Now we got into the hassle of whether it was the toner (said the photocopy repairman) or the photocopier (said the company that sold the toner to us) causing the problem. Is this starting to sound familiar?

We finally bought some Canon® toner from our local office supply store at list price. It was \$100 per box. **Ripped-Off!**


The question is, "Why was I ripped-off?" The answer — because I was trying to save a fast buck. I tried to beat the normal pricing system by getting a deal. Instead, I have almost two boxes of worthless toner. All rip-off schemes are based on the fact that people are always looking for the fast buck. As long as they keep looking, there will be rip-off artists.

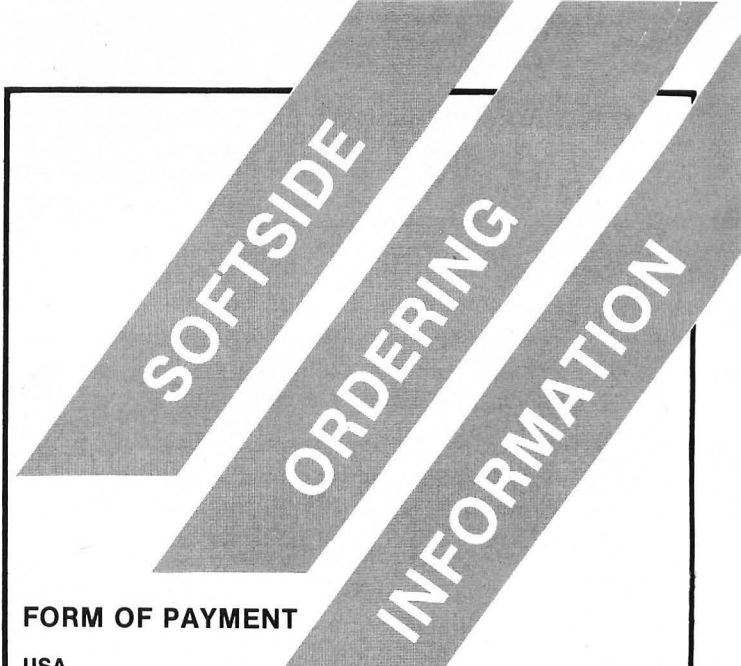
One rip-off I ran into several months ago was over my own *ST80-III*™. It seems a computer store in Canada was having a going out of business sale. They had a terrific deal on *ST80-III*™ — only \$75.00. I know of at least 30 people who bought their *ST80-III*™ from this store. It is now out of business and so are the people who bought my software from them. In the first place, all 30 of these *ST80-III*™'s have the same serial number. The instructions were obviously photocopies and not everybody got all of the instructions. Some people didn't get any instructions at all.

Some of these people are trying to smile about it, since nobody wants to admit paying \$75.00 for pirated software. Everyone knows you can get pirated software for free. Only a jerk pays for it. A few have come to me and now expect me to bail them out. They say that they paid for an *ST80-III*™, they didn't take a free pirated copy. They were being honest with me. Shouldn't I help them?

I didn't rip them off. In fact, I never got a red cent of their money. If they had wanted a legitimate copy of *ST80-III*™, why didn't they buy it from a reputable dealer? If they didn't know any reputable dealers, why didn't they buy it directly from me? The answer is simple. They were trying to save a fast couple of bucks and they got caught.

It's just like my "California toner." What would my local office supply store have said had I asked them to replace the cheap toner I had bought from a place at the other end of the country? They would have said that they didn't sell me the stuff and didn't rip me off. Somebody else did. If I had wanted good toner, why didn't I buy it from them? Heck, they'd have given me a better price.

The reason a lot of computer users are being cheated is because they, themselves, are cheating by trying to avoid paying a fair price for good products. In the end, it's only going to cost them more money. There are no "great" deals in this industry. So when you don't find them, remember, it's the consumer who pays the fair price who gets the "fair" deal. 



FORM OF PAYMENT

USA

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

Canada/Mexico

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

Other Foreign Orders

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

GUARANTEE

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

LIABILITY

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

PRICES

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

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

	USA/Canada APO/FPO	USA/Canada FIRST CLASS Mexico	Other Foreign
SoftSide Magazine (yr)	\$30	\$40	\$62
	USA APO/FPO	Mexico Canada	Other Foreign
CV (year) & magazine (6 mo.)	\$75 \$39	\$95 n/a	\$125 n/a
DV (year) & magazine (6 mo.)	\$125 \$64	\$145 n/a	\$175 n/a
Adventure of the Month Month (6 mo.)			
Cassette	\$29	\$35	\$41
Disk	\$49	\$55	\$61

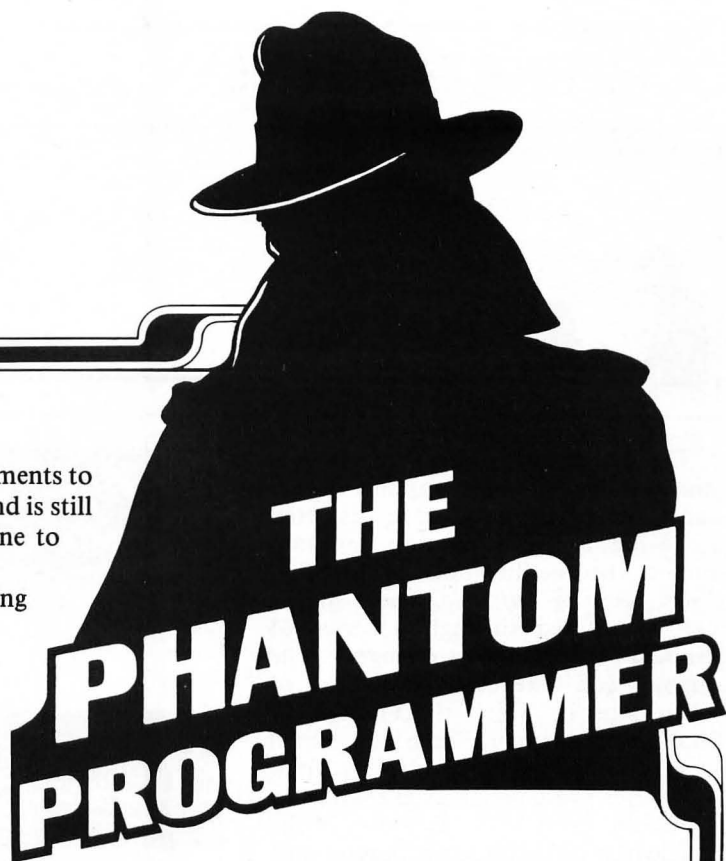
BACK ISSUES

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

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

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

A FEW WORDS FROM...



A NEW COMMAND FOR BASIC.

There have been a number of utilities for and enhancements to BASIC published in recent months. One crucial command is still lacking, however, and I feel that it is time for someone to speak out on the problem.

Anyone who has done any amount of programming knows how easy it is to get carried away. No one truly knows how many a poor soul has sat down at the keyboard to accomplish some small task, only to later find that when he looks up at his wife, she's left for mother's, his leg muscles have atrophied, and his home has been boarded up around him.

Most set out with the best intentions, too: "I'll just finish the printer routine and then go to bed."

"Just that one bug to find and I'm finished."

"One more game and then I've got to wrap it up."

Well, we all know about good intentions. There is an answer, however. It's called PUNT. PUNT is a command which wrests control of the system from the operator when some preset condition is met. The routine then zeros all of core and locks up the keyboard. The only escape is to power down the system for a period of time equal to one night's sleep.

As far as I know, no system offers a true PUNT. I am told that the ATARI® already supports a primitive form of PUNT. The ATARI® 800 allegedly locks itself up from time to time, requiring that the system be shut down and cold-started. This version is not to be recommended, however, as it seems that not even ATARI® knows how it works or how to control it.

I am also painfully aware that TRS-80® Model I's will spontaneously reboot and do other interesting things without warning. Really, though, this could scarcely be considered to have the same vigor and elegance as PUNT.

As a BASIC operator, PUNT can be called under a wide variety of conditions. For instance, you might wish to read the real-time clock and set a limit:

```
10000 IF TIME$ = "03:00:00" THEN PUNT : REM 3:00 AM
```

Or, perhaps you might use some variable to record the number of revisions to the current program you have saved in this session and set a limit there:

```
10000 NR = NR + 1 : IF NR > 20 THEN PUNT
```

Direct Machine-Language calls to the PUNT routine could be used to monitor error counts in programs like SCRIPSIT (which I am using to write this). Such calls could also keep an eye on other programs which do not run directly under BASIC.

Other possibilities suggest themselves. If your wife is pounding on the door to the computer room, this could be wired to be a hardware interrupt and could be detected by appropriate sensors connected to an input port. If she were yelling at you, this would be a software interrupt and could be similarly handled. In either case, it might be wise to PUNT.

Then again, you could (Yawwwnnnn....)mmm, excuse me. It's getting late. Well, as I said, there are many ways in which this useful and necessary command could improve the lot of programmers everywhere. All that is ne ***PUNT***



Transformation Techniques:

Variation in the Rate of Change

by Joan Truckenbrod

The shape of objects or figures drawn on a computer graphics system can be changed, modified or distorted by using a transformation program such as the one illustrated in the May, 1982, issue of *SoftSide*. These shapes can be altered through a series of gradual or dynamic changes. The changes can take place in a series of even steps in which the amount of change is the same for each step. Equal increments of change in a transformation series produce an even rhythm or a smooth visual flow, as shown in the article in the May issue. The amount of change in the shape of the figure from one step to the next in a transformation series can vary within that series. This article discusses a technique for varying the amount of change within a transformation sequence to create various visual effects. For example, the change in shape at the beginning of a sequence can be very subtle, gradually increasing until the change in shape in each step of the sequence is dramatic. Thus, the shape or figure will appear to change slowly at the beginning of the sequence, and much more dynamically at its end.

The rate of change in a transformation series can create a variety of visual effects. The series of shapes in Fig. 1 shows the transformation of the beginning shape into the end shape using a varied rate of change. At the beginning of the series, the shape changes slowly. You can see that the shapes on the left side of the series are very similar. The amount of change between the figures gradually increases and consequently the figures near the end of the series are very different from one another. Compare this series with the series of shapes in Fig. 1 in the May article. In that case, the amount of change between each step is equal. When equal intervals of change are used, the middle shape in this series is a square. As shown in Figure 1 in this article, progressive intervals provide a different series of shapes. The series appears to begin slowly and then move more rapidly at the end of the sequence. This

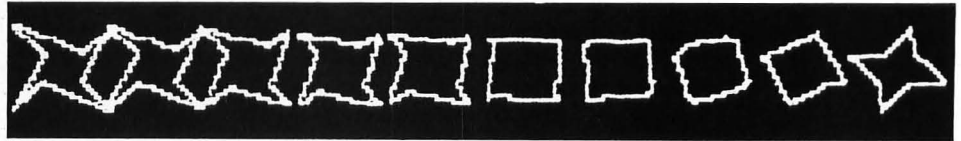


Figure 1

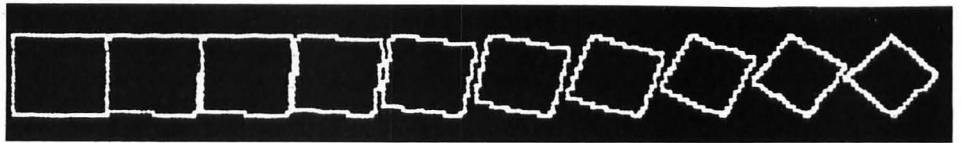


Figure 2a

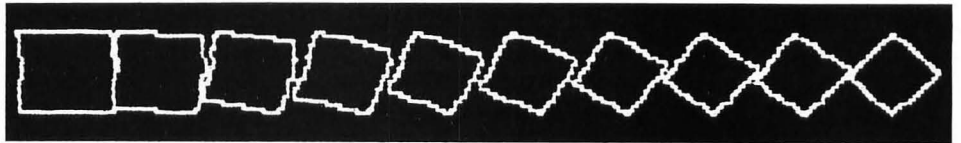


Figure 2b

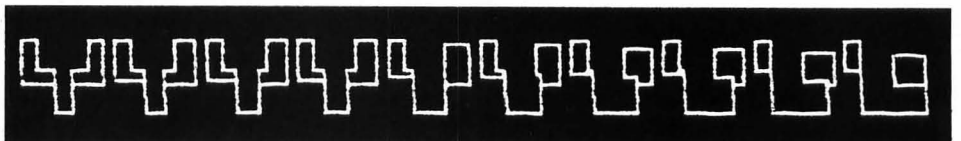


Figure 3

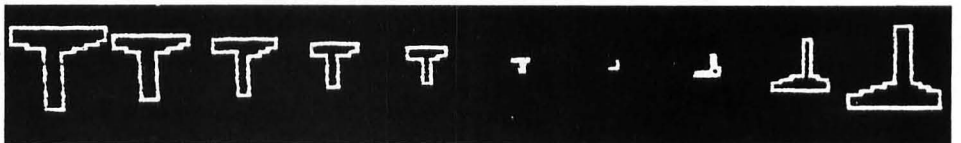


Figure 4a

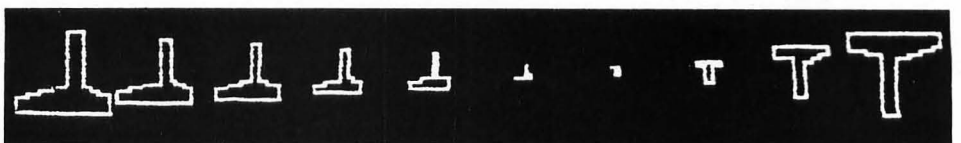


Figure 4b

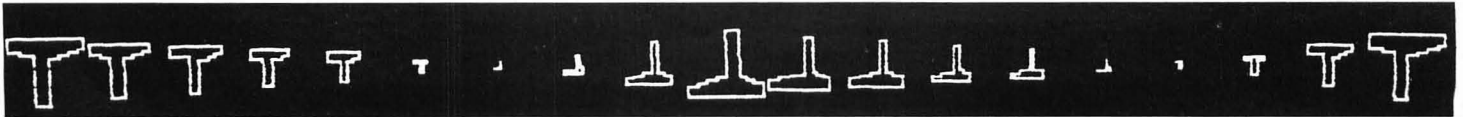


Figure 4c



Figure 5

concept of speed is clarified in the comparison of Figures 2a and 2b. Both series transform a square into a diamond. However, the rate of change is different in each. The series in Fig. 2a begins with a small amount of change between steps, which gradually increases towards the end of the series. Fig. 2b illustrates the opposite as the series begins with a larger amount of change between the figures, decreasing as the series progresses. The visual speed, or acceleration rate, in a series of figures or animation is determined by the rate of change in the shape of a figure. The visual appearance of varying speeds is shown in Figures 4a and 4b. In Figure 4a, the letter T appears to gradually move back into space, flip upside down, and then move rapidly out towards the viewer. The difference in the speed of the figure moving into space and of the figure moving out towards the viewer is dependent on the amount of change between the figures in the sequence. The greater the amount of change between the figures in a series, the faster the speed of the figures appears to be. Figure 4c is a combination of Figures 4a and 4b and shows the figure moving in and out of space at varying rates. Figure 5 also shows a combination of two series in which a horizontal bar is proportionally transformed into a vertical bar. In the left side of this diagram, the amount of change between the figures gradually increases from a small amount to a larger amount. Then, beginning at the center vertical bar, the transformation begins with a small amount of change that gradually increases towards the end of the series. Since the diagram in Figure 5 is not symmetrical, it is clear that varying the rate of change in a transformation series has an effect on the progression of its shapes.

In a regular transformation, the X and Y coordinates that define the initial figure are continually changed by a consistent amount, determined by the transformation program. To create variation in the rate of change between the steps in a transformation series, the amount of change in the X and Y coordinates is determined by a series of pro-

portionally related numbers, provided by the programmer. This series of numbers represents the relative amount of change that is to occur in the figure in each successive step of the series. This list of numbers is put into the PC array in the DATA statement in line 105 of the program below. The PC array is DIMENSIONED according to the number of desired steps in the transformation sequence. The series of numbers used in this program was chosen to produce a gradual change in the beginning of the series, increasing to produce a greater amount of change in the figure at the end of the sequence. Experiment with various series of numbers in the PC array, creating a variety of rhythms in transformation sequences. Studies with different visual speeds and acceleration rates can be enhanced by increasing the number of steps in the transformation sequence.

```

10 REM SHAPE TRANSFORMATION
15 REM VARIATION IN SPEED
20 REM QF CHANGE
25 REM
30 REM BY JOAN TRUCKENBROD
35 REM
40 NS = 8: REM NUMBER OF STEPS
   DESIRED BETWEEN BEGINNING
   AND ENDING FIGURE
50 NS = NS + 1: REM ACTUAL NUMBER
   OF TIMES X AND Y COORDINATES
   ARE INCREMENTED FOR THE
   TRANSFORMATION SERIES
55 NP = 9: REM THE NUMBER OF
   POINTS USED TO DEFINE EACH
   FIGURE
57 REM
60 DIM X1(NP),Y1(NP): REM COORD-
   INATES FOR BEGINNING FIGURE
70 DIM X2(NP),Y2(NP) REM COORD-
   INATES FOR ENDING FIGURE
80 DIM CX(NP),CY(NP): REM DIS-
   TANCES BETWEEN CORRESPONDING
   POINTS IN PAIR OF FIGURES
90 DIM PC(NS): REM PERCENTAGE
   OF CHANGE FOR EACH FIGURE IN
   TRANSFORMATION SERIES
95 REM

```

```

100 FOR I = 1 TO NS: READ PC(I):
   NEXT I
105 DATA .12,.16,.22,.30,.40,.54
   ,.65,.82,1
110 FOR I = 1 TO NP: READ X1(I),
   Y1(I): NEXT I
120 DATA 0,50,16,58,24,50,16,66,
   24,74,8,66,0,74,8,58,0,50
130 FOR I = 1 TO NP: READ X2(I),
   Y2(I): NEXT I
140 DATA 8,58,16,50,16,58,24,66
   ,16,66,8,74,8,66,0,58,8,58
145 REM
146 REM THE FOLLOWING LOOP DETER-
   MINES THE DISTANCE BETWEEN
   EACH PAIR OF CORRESPONDING
   POINTS IN FIGURES ONE AND
   TWO
150 FOR I = 1 TO NP
155 REM CX AND CY REPRESENT THE
   X AND Y DISTANCE BETWEEN A
   PAIR OF CORRESPONDING POINTS
   IN FIGURES ONE AND TWO
160 CX(I) = X2(I) - X1(I)
170 CY(I) = Y2(I) - Y1(I)
200 NEXT I
210 REM
220 HGR : HCOLOR= 7
225 REM
230 REM THE FOLLOWING LOOP DRAWS
   THE SERIES OF TRANSFORMED
   FIGURES
240 REM N REPRESENTS THE NUMBER
   OF FIGURES IN THE SERIES AND
   M REPRESENTS THE NUMBER OF
   POINTS IN EACH FIGURE
250 FOR N = 0 TO NS
260 XP = X1(1) + N * 23 + (CX(1) *
   PC(N))
270 YP = Y1(1) + (CY(1) * PC(N))
280 HPLLOT XP,YP
290 FOR M = 2 TO NP
300 XP = X1(M) + N * 23 + (CX(M) *
   PC(N))
310 YP = Y1(M) + (CY(M) * PC(N))
320 HPLLOT TO XP,YP
330 NEXT M
350 NEXT N
400 END

```



Why Johnny Can't Program - Materials for Computer Literacy

by Dean F. Hayden Macy

Since the beginning of time, human beings have been at war. It started in the Garden with the battle between good and evil. That battle continued on until there was a disagreement between two people on the same side. In our complex society, war has evolved to a fine art. Today we tend more to the alienation of people and the alienation of minds. The largest, single battle today is being waged over education — not whether it should exist or not, but how. The combatants I'll address here are the computerists and the anti-computerists. On the computer side are educators, teachers and students. Equally biased on the other side are, again, students, educators and teachers.

David Moursund, Professor of Computer and Information Science at the University of Oregon says, "Our children will need a functional knowledge of computers to compete in the future job market. By 1985 a high percentage of all jobs will involve computer use in some way, and those who don't know how to use them will be at a definite disadvantage. The educational system has not kept up with this trend because not enough teachers are computer-literate and because there are not enough computer related instructional materials. The ability to use computers could become as important as the ability to read and write. Our educational system is completely dependent, in essence, on how teachers and parents view the change in the world."

Some elementary and secondary teachers are against using computers in the schools. They see the change as a threat to their livelihood. These teachers feel that computers will replace them; that they will be relegated to the inane tasks of taking attendance and monitoring student needs.

I can't imagine teaching has changed much since I attended school. In the

classroom, 70% of a teacher's time was spent controlling the unruly. The remaining time was used for teaching and administration. Gifted children were lost in the shuffle. Many lost their incentive to learn, and some of these, to this day, are wandering around in our complex, technical world performing menial tasks. "A mind is a terrible thing to waste."

"The computer issue is not merely a theoretical problem posed by educators. Real obstacles must be overcome if students and teachers are to truly benefit from the classroom computer. It is important to realize the computer is not replacing the teacher, but is only assisting the teacher to educate the student. Teachers will still teach. Microcomputers have limitations, so the need for adaptable, responsive teachers will always be present." James Keogh, educational advisor, stresses that teachers who embrace the computer must be adaptable; willing to change the way they think. Whereas children seem to have no problem adapting to change, the older one gets, the less one is willing to embrace change. With the onspread of computer games in homes and arcades, children are primed for computer learning, but teachers, even those generally not concerned about being replaced by the computer, are still apprehensive.

Why? The biggest reason is the quality of available computer software. Most "educational" programs are not created with education in mind. Programmers, attempting to gain entrance into the field, often take a program designed for gaming, change a few lines, and label it "educational." Although this modified program may be extremely well structured, it may not be user-friendly. In fact, it may be very hostile. It is crucially important that programs, written to be used in the field of education, be extremely friendly to the user. *K.I.S.S.* (Keep it simple,

stupid.) should be the educational programmer's motto. Programmers need to err on the side of simplicity, settling for modest goals in terms of explanation, or the amount of information conveyed. Programs are successful when they can give large audiences a learning experience they will enjoy and motivate them to go further into that experience.

James Keogh says, "Educational programs must be challenging for students and meet their educational needs. Programs must be able to co-exist with current learning methods and textbooks. Programmers must anticipate every possible problem that might arise during a learning session, and develop contingencies for them. All ambiguous words and phrases must be removed and the software must be very carefully assembled and debugged; if not, the student and the teacher will become very frustrated."

Frustration is commonplace for reviewers of educational software. A new educational monthly disk/magazine for the Apple™, *WINDOW*, touted to be the last word in computer learning, contains many bugs and is not very user-friendly. *WINDOW* could be made much more user-friendly by trapping errors. Instead of the programs crashing, a message could be flashed to the user, prompting him/her to reenter the program using XXX key.

I have found many pseudo educational programs which talk down to the student with demeaning statements. Imagine a student, upon three wrong responses in a computer administered test, getting a "raspberry" accompanied by a CRT message saying, "OK Stupid! Obviously you have not studied your textbook. I don't work with dummies," and the screen goes blank. (Yes! It is a marketed program.) At first glance, the statement is humorous, but after a moment, the full import of the message sinks home and

“The ability to use computers could become as important as the ability to read and write.”

David Moursund, Professor of Computer & Information Science at the University of Oregon.

you become angry and frustrated; angrier still if you're the one at whom the message is aimed.

Another, more common problem, is English usage and mathematical accuracy. Many, otherwise excellent, programs have severe vocabulary problems. Programmers tend to promote the same program to tutor elementary and college students, changing only the level of the utilized facts, disregarding the context. Thus, a second grader may be asked to decipher a question such as, “What is the potential of technological advancement in today's complex society?”; which is not quite as funny as the college student who is asked, “See robot run? Run robot, run. See the space ships? The robot runs to the ships. How does the robot get into the ship?”. Mathematical problems prove to be annoying to teachers when compiled by the programmer. Assume the problem: 2 plus 3 times 4. In the lower grades, children studying arithmetic will answer 20 and be correct. But, a high school student studying algebra will answer 14 ($2 + (3 \times 4)$). Both students' answers are correct, however most programmers do not allow for varying correct answers to a question.

Of the nation's 1,085,000 educators, how many are computer aware? *Instructor*, the national education magazine, ran a Computer Attitude Survey. The results are telling. Of the 4000 classroom teachers who responded, 86 percent expressed a high level of interest in computers, however only 39 percent actually used computers for instruction; fortunately 38 percent had taken inservice courses in computers. The survey also showed that twice as many students as teachers had participated in real-time experience working with computers. Educators reported Apple™ the most used school computer with TRS-80® the runner-up. ATARI® rated fifth. These computers were used primarily for drill practice and enrichment. 40



percent of the school computer's time is devoted to programming. One reason is clear; software available to the teacher is not designed for classroom use, and programs designed for the classroom are, for the most part, not properly focused for the intended age group. The ultimate solution is to have a child write software for children, but most children with the awareness to program a computer don't have the educational background to write the required software.

During the weeks of preparation for this article, I was inundated with texts and software of every description by many authors. Some were educators, many were teachers and a few were mothers and fathers of questing children. By far, the material I read authored by parents used a more direct approach to children's needs than most of the writings and programs offered by the professionals. If educators would remember their own childhood, the approach to computer awareness materials for children would be much different.

The child always wants the dessert first. He doesn't care about the history and technological development of the computer. He does not want to know about the hardware governing the operation of the computer. He cares even less about Boolean algebra or binary numbers. (Would you care if you were a child?) He may be interested, to some minor degree, in how a computer “thinks,” but what he really wants to do is turn the thing on and watch it make pictures or do something constructive. After that, he wants to play with the keyboard and make the computer do “tricks.” Reasoning as a child, an author with some programming skills could write a manual to teach children to program a computer. If the author wanted the child to have some of the background mentioned above, he could offer a taste of the dessert, holding back the remainder until the child had digested the liver. (Ugh!)

Tandy Corporation markets TRS-80® Computers. Their Model I manual assumes nothing about the reader other

than that he or she can read. This is an ideal learning tool for children of any age who want to study programming from the beginning. The author takes a person from the fundamentals and eases him through to advanced programming skills, without fanfare, and using a little tongue in cheek humor to help lighten the load. At first glance, it looks as though the manual was written for idiots. However, aren't we all idiots when the time comes to learn something we know absolutely nothing about? If other computer manufacturers wrote instruction manuals along the same lines as the one offered by Tandy, there would be little need for the hundreds of "how to" books littering my desk. Alas, the authors of the Apple™ and ATARI® manuals were not as aware of their audience.

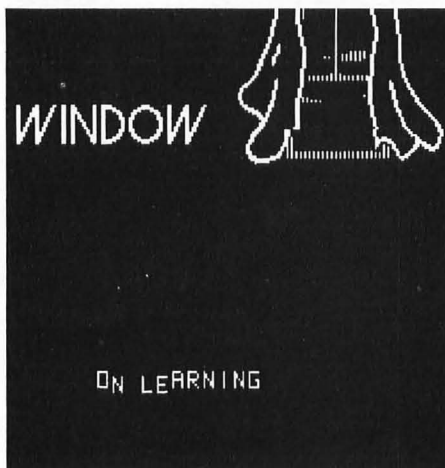
Camelot Publishing is a distribution company for computer awareness books and materials geared for the educational field, written by Donald D. Spencer, Ph.D.. Using an offbeat approach to programming and computer instruction, he writes excellent material for the school child, in a manner the child can accept. (He remembered that he too was once a child.) Of over 60 books Camelot makes available to the elementary teacher, I reviewed ten. The books *Fun With Microcomputers and BASIC*, *Visual Masters for Teaching Basic Programming* and *Computer Awareness Coloring Book* proved to be the best of the lot. After reading *Fun With Microcomputers and BASIC*, I wholeheartedly agreed with *The Mathematics Teacher* editorial which stated, "This book is written in a humorous tone with many programs, activities, and illustrations that portray the computer as a "fun" machine — something to be enjoyed and used for solving problems." Spencer is a fun person and a prolific writer who is able to ease a whopping amount of data into a child without the child ever realizing he has assimilated the amount of information fed to him. (To obtain a catalog of Spencer's materials write to: Camelot Publishing Company, P.O. Box 1357, Ormond Beach, FL 32074)

For the very young child, the book from Creative Computing Press, *Katie and The Computer*, by Fred D'Ignazio, is a charming and effective teaching tool. Creative Computing also has a few books authored by Debbie Larsen, a talented mother, who has taught her own children to program an Apple™ by presenting data to the children in a way they can understand. (The people at Apple™ publish many excellent books, but most are far beyond the grasp of young people and

some adults as well.)

The only other teaching manual I reviewed which I would recommend is written for the Apple™. A tome by Aubrey B. Jones, Jr., it's entitled, *I Speak Basic To My Apple*. Whereas the book, in my opinion, addresses the liver before the dessert, it is nevertheless an excellent teaching tool for the Junior High School student. The manual assumes the teacher knows nothing about the Apple™ computer, but always keeps the instructor ahead of the class. (available from Hayden Publishing Co.)

For a little Apple™ polishing, try *WINDOW*. Although the initial disk contains bugs and is not very user-friendly, one program written for the beginning programmer shows how any



short, user-entered program, containing INPUT, FOR-NEXT loops, and GOSUB, functions during real-time use. The student can step through his program, one instruction at a time, or crawl through the address lines in slow motion.

If I were to write a book on programming for children, I would first let the child taste the dessert, by demonstrating a simple program on a CRT readout. Next, I would guide the student through the computer's "thinking process." Using everyday examples, such as eating or opening a door, I would have the child trace, in block diagram format, everything a computer controlled robot would do to mimic the human movements. I could then show pictures, explaining the workings of a player-piano, and how it is programmed, comparing it to the computer. (The liver, so to speak.) After this short introduction was digested, I could give the child his dessert by having him perform a simple programming exercise using a micro-

computer keyboard. This done I would then.....Oh No! The BBC pre-empted me. (Great minds and all that!)

Last evening I was given a video cassette tape and asked to play it over my "telly." It was an offering from the BBC entitled *The Computer Programme*, which eases you into computer literacy using exciting video techniques. The tape I viewed was the third in a series of ten, twenty-five minute programs offering an in-depth exploration of the microcomputer. "Talking To A Machine" not only stole my ideas on how to effectively present computer programming to a child; it did it in grand style. The magnificent Trooping the Colour ceremony was used to illustrate, with royal soldiers marching in precision to a simple command, how a computer accepts commands and acts accordingly. And an ancient, working steam organ played melodies reminiscent of the calliope of the good ole' days, using a punch card format, not unlike IBM cards, to feed the data. (A player calliope, if you will.) The program is presented in the relaxed British style and is entertaining as well as instructive. The series is available in a multitude of video tape and film formats priced from \$1700, truly a low cost program for a school or library, considering the educational content. (For more information on this thrilling series, contact: Films, Inc., 733 Green Bay Road, Wilmette, IL 60091.)

Following a recent study of computer literacy among high school graduates it was found that few graduates ever see a computer in the classroom. They have no concept of the computer's effect on society in the future. But, this will change and is changing.

It is clear that computerized education is inevitable. It will be vastly better than the present system, far more efficient, and *fun*. The single key to education is that learning *must be fun!* Education courses will be presented by enthusiastic people in interesting ways. More Spencers and Larsens will graduate from school systems presently using computers in education and new ways of teaching will emerge. Education will expose our impressionable children to the fascination of electronics, communications and computers, and the world will change. It won't be the world of Orwell's *1984*, but a world where challenges will be embraced, children will not lose their incentive, identities will not be lost in the shuffle — a world where a child's brain can't be wasted.

The world I want to live to see will be *that world!*

CATS:

Computer Assisted Testing System

PART ONE

by Jon R. Voskuil
ATARI® and TRS-80® translations by
Alan J. Zett

CATS is an educational utility program for an Apple™ with Applesoft and 32K RAM (tape) or 48K RAM (disk); an ATARI® 400/800 with 16K RAM (tape) or 24K RAM (disk); or a TRS-80® Model I/III with 16K RAM (tape) or 32K RAM (disk).



With an increasing number of computers finding their way into schools and homes, teachers and parents are looking for educational ways to use them. *CATS* is a set of programs that will enable you to use your computer to create and administer tests, and to keep records on the results of the testing. This first module is the one necessary to create the tests; the test-giving and record-keeping modules will follow in the next issue of *SoftSide*.

The test creation program operates from a main menu which enumerates all of the options and is generally self-explanatory:

1. Add to test in memory
2. Review test in memory
3. Save test to disk (tape)
4. Begin a new test
5. Load a test from disk (tape)
6. Reset question entry options
7. Print test to printer
8. Quit

When running the program for the first time, the only logical choice is to begin a new test, option 4. In the disk version, you will be prompted for a file name to use for your test, and your disk will be checked to make sure that you won't be writing over an existing file without knowing it. The tape version skips directly to the next part.

You will then be asked to choose among a number of options to be used during the creation of the test. Three types of questions may be entered: true/false, multiple-choice, and fill-in-the-blank. You may choose to have all questions of the same type, or to intermix the types throughout the test. You may then choose either to give the same value to all the questions, or to enter the relative value of each question separately. Further, you may elect to use a hint mode which allows you to enter a hint with any of the questions, with an optional corresponding penalty if the student asks for the hint. If you turn the hint option on, you are not required to enter hints, but you may if you so desire. You can also specify a uniform penalty for all hints, or choose to give each hint its own penalty factor. The penalty is expressed by a number from 0 through 9, representing the number of tenths of the question's value to be deducted for using the hint.

Any of these options may be reset during the creation of the test simply by returning to the main menu and choosing option 6. For example, this allows you to enter a number of true/false questions in a row, and then switch to multiple-choice; or to give a uniform point value to a group of questions, and then enter variable values for others.

Once these options are chosen, actual entry of the questions and answers

begins. The design of the program limits the text of any single question and this is somewhat different for each system. On the Apple™, the limit is 230 characters, or about six screen lines. On the TRS-80®, the limit is also 230 characters. This is a little more than three and a half screen lines. The ATARI® version allows you one full screen line or 38 characters. This is enough for most needs. If not, the limit may be raised by changing the value of LN in line 1. If you reach the limit, the computer will not allow you to enter more characters. The Apple™ and ATARI® version will also beep at you. Any characters may be typed into the text, including all punctuation. Quotation marks, however, will be converted to apostrophes (single quotes) because of disk file format.

As you type in a question, you may backspace in the normal way to correct mistakes. If you want to erase what you've entered so far and start at the beginning, pressing CTRL-E will do the trick. (It's much faster than backspacing to the beginning.) If you have occasion to enter underline characters into the text (for fill-in-the-blank questions, for example), you may do so using CTRL-U. Pressing RETURN signals the end of the input, in the usual way. If you press RETURN only when prompted for the text of the question, you will be returned to the main menu. Keep in mind

that the question will be displayed exactly as you type it on the screen, so be careful to format it neatly in the available space, avoiding messy end-of-line breaks.

Depending on the type of question you are entering, and the options you have chosen, you will then be prompted for the possible and the correct answers, the question value, the hint, and the hint penalty. Since true/false questions always have the same answer choices, you need only enter T or F for them. In the case of multiple-choice questions, you must enter all the choices, pressing RETURN after each one and pressing it once more following the last choice. You are then prompted for the number of the correct choice (one only). In the case of fill-in questions, you may enter more than one correct response if you so desire. You may also choose how many of the initial characters (1-9) of the response must be matched in order to be judged correct; pressing RETURN only will require a perfect match.

At any time, the test being created may be saved. Return to the main menu by pressing RETURN when prompted for the next question, and then choose option 3. At this point, the disk version will give you the option of changing the disk file name, or using the one already entered. After saving it, you can then continue by choosing option 1.

You may also review the questions entered so far by choosing option 2. This will display each question in turn, along with all possible and correct answers, hints, values, and penalties. Multiple-choice questions will have the correct answer number displayed in inverse on the Apple™ and ATARI®, and marked with an asterisk on the TRS-80®. Fill-in questions will have the number of characters required for each answer specified on the ATARI® and TRS-80® and displayed in inverse on the Apple™. (A later addition to this part of the program will allow you to edit questions as well as simply review them.)

Once you have a test or two saved, you can recall them at any time using option 5 from the main menu — either immediately upon running the program, or at any later time. If you already have a test in memory, you will be asked to confirm that you want to load a different one, since the current one will be wiped out. (The same confirmation is required for option 4, beginning a new test.)

Finally, the printout option (number 7) will provide a hard copy of the test in memory. The Apple™ version

assumes that your printer controller card is in slot 1. The printout routine also assumes that you'll be using the printout only for proofreading or record-keeping purposes: the text is formatted for the screen, and will look a little odd on an 80-column printer.

The next CATS module will enable you to administer, to individual students, the tests you have created and stored. It will include features such as administering all or just part of a test; enabling or overriding various options that you have already included in the created test; providing various levels of feedback to the student about his scoring; and security protection, so that students will not easily be able to access the correct answers or other students' scores.

Programming Notes

The discussion which follows refers primarily to the Apple™ and TRS-80® versions. Since there are no string arrays in ATARI® BASIC, the question, hint, and answer data are stored as parts of three very large single strings rather than as elements of string arrays. Also, the digits referred to in the discussion are appended to the end of each data string rather than the beginning in the ATARI® version.

The Q\$ array holds the text of each question; the H\$ array holds the corresponding hint, if any; and the A\$ array holds all the answers for the multiple-choice and fill-in questions. In addition to the actual question text, each entry in the Q\$ array has concatenated onto the beginning of it several numbers which represent that question's parameters. True/false questions have five such digits, and the other two types have nine; they are concatenated in lines 3500 and following.

The second digit represents the question's relative value, which defaults to 1 if no individual values are assigned. The last two digits prior to the text of the question are also the same for all question types: The next-to-last represents the hint status (0 is off, 1 is on), and the last represents the hint penalty.

Between the first two and last two digits, the question types differ in their format. For true/false questions, only one digit is needed here, representing the correct answer: 1 for true, 0 for false. The other two question types use a total of five digits to point to the possible and the correct answers. First there is a three-digit pointer to an element of the A\$ array, which is either the first answer choice (in the case of a multiple-choice question) or the first

correct response (in the case of a fill-in question). The next digit tells how many possible (multiple-choice) or correct (fill-in) responses there are in the A\$ array for that question. And the last of these five digits gives either the number of the correct answer (multiple-choice) or the number of characters that must be matched for a correct response (fill-in).

The subroutine at line 15000 of the Apple™ disk version is the one found in the *Applesoft Reference Manual* for fixing problems caused by the use of the ONERR GOTO statement. This is poked into memory at the beginning of the program, and then called whenever an ONERR statement has been executed. Without it, such things as GOSUBs and RETURNS can get quite messy.

The routine at line 15000 of the Apple™ tape version is a routine to read and write string arrays to and from tape. The routine is poked into memory at the beginning of the program and called whenever a test is loaded from or saved to tape. This is the same routine that was printed in the May, 1982 issue of *SoftSide* as an enhancement to *Microtext 1.2*.

In the Apple™ version, the "Wait for a keypress" subroutine at line 10000 uses Applesoft's WAIT statement, a very handy but little used feature. It can be used to wait for a specific keypress (or other, more esoteric functions), but in this case simply waits for ANY key to be pressed. Using the mnemonic variable names KEY and PRESS makes the statement speak for itself very nicely.

The menu formatting subroutine at line 11000 is one which can be easily transplanted to other programs. It can be used either in conjunction with DATA statements to READ the necessary menu information, or with string assignment statements. Both methods are used in the CATS program. When DATA items are to be READ, the routine is entered at line 11000; when string assignments have already been made, the GOSUB is to line 11010 instead.

A NOTE ABOUT THE LISTINGS:

In the listings which follow, the complete disk version of the program is listed first. The changes necessary to use CATS with tape are printed following the disk version listings. The SWAT table for the tape version is for the complete tape program and assumes that the user has entered the disk version and then made all the changes exactly as specified.

Disk version variables:

A\$: Answers to questions. (ATARI)
 A\$(*): Array of answers to questions. (Apple/TRS-80)
 A1: Pointer to an answer number.
 AP: Pointer to next answer number (while building array of answers).
 B\$: Backspace character.
 BEL\$: Bell character. (Apple/ATARI)
 BR\$: String containing left and right bracket characters. (Apple)
 C: Correct answer number, or number of correct answers (depending on question type).
 C\$: Input character.
 CC: Value of input character.
 CLR: Keyboard clear address. (Apple)
 CM: Number of characters to match.
 D\$: Control-D. (Apple)
 E: Error code. (Apple/TRS-80)
 EO: Entry options flag. Equals zero if options have not been chosen.
 F\$: Test name.
 FIX: Address of ONERR-FIX routine in memory. (Apple)
 H: Used for horizontal positioning.
 H\$: String of hints. (ATARI)
 H\$(*): Array of hints. (Apple/TRS-80)
 HP: Hint penalty.
 HS: Hint status.
 HYNT: Hint flag.
 I: Loop variable.
 I\$: Input string.
 IT: Value of input character.
 J, JJ: Loop variable.
 KEY: Address of keyboard strobe. (Apple)
 L, N: Misc.
 N\$: Menu title. (Apple/TRS-80)
 N\$(*): Menu options. (Apple/TRS-80)
 NC: Number of characters to match.
 NN: Number of correct answers.
 PFLAG: Hint penalty flag.
 PRESS: Equals 128. (Apple)
 PWS\$: Test file password.
 Q\$: A question. (Apple/TRS-80)
 Q\$: All test questions. (ATARI)
 Q\$(*): Array of test questions. (Apple/TRS-80)
 QF\$: Disk file name of test.
 QN: Question number or total number of questions.
 QUOS\$: Equal to a quotation mark. (Apple/TRS-80)
 QV: Question value.
 S: Address of speaker. (Apple/ATARI)
 S\$: String of spaces. (Apple)
 S\$: Delete line character. (ATARI)
 T: Question type.
 T\$: Question type names. (ATARI)

T\$(*): Array of question type names. (Apple/TRS-80)
 TF: True/False flag.
 TF\$: Contains "TRUE" and "FALSE". (ATARI)
 TF\$(*): Contains "TRUE" and "FALSE". (Apple/TRS-80)
 TYPE: Question type.
 U\$: Underline character. (Apple)
 U3\$: Three underline characters. (Apple/TRS-80)
 V: Used for vertical positioning.
 VFLAG: Question value flag (fixed or variable).
 VV: Used for vertical positioning.
 X: Misc.
 X\$: Single character input string. (Apple/TRS-80)
 Z: Loop variable.

Additional tape version variables:

RD: Address for tape read routine, equals 800. (Apple)
 WR: Address for tape write routine, equals 768. (Apple)



```

#####
$      ATARI BASIC      $
$      "CATS"          $
$  Author: Jon R. Voskuil $
$  Translated: Alan J. Zett $
$      (c) 1982, SoftSide $
#####
  
```

Set question line length.

```

1 LN=1
10 GOTO 500
  
```

Custom line input routine.

```

100 I$="" : V=PEEK(NB4) : H=PEEK(85) : POKE
CRS, NO : ? CHR$(254) ;
110 GET #N1, C : IF C<32 OR (C>122 AND C<
160) THEN 150
130 C$=CHR$(C) : L=LEN(I$) : IF L=LN*N38 O
R (L=N38 AND NQ=N1) THEN ? CHR$(253) ; :
GOTO N110
140 I$(LEN(I$)+N1)=C$ : ? C$ : GOTO N110
  
```

Process control commands.

```

150 IF C=155 THEN POKE CRS, N1 : NQ=N1 : RE
TURN
160 IF C=21 THEN C=95 : GOTO 130
170 IF C=N5 THEN GOSUB N1000*N10 : GOTO
N100
180 IF C<>126 THEN GOTO N110
190 C$=CHR$(C) : L=LEN(I$) : IF L<N2 THEN
  
```

```

I$="" : GOTO 210
200 I$=I$(N1, L-N1)
210 IF L>NO THEN ? C$ ; " " ; C$ ;
220 GOTO N110
  
```

Define variables.

```

500 N0=0 : N1=1 : N2=2 : N3=3 : N4=4 : N5=5 : N7=7
:N9=9 : N10=10 : N14=14 : N15=15 : N37=37 : N38=
38 : N39=39 : N48=48 : N84=84 : N89=89
502 N100=100 : N107=107 : N110=110 : N1000=1
000 : N11000=11000 : CRS=752
  
```

Display title page and initialize program.

```

510 GRAPHICS NO : POKE CRS, N1 : CLOSE #1 : O
PEN #N1, N4, NO, "K"
520 POSITION 17, N5 : ? "C A T S"
530 POSITION 12, N7 : ? "by Jon R. Voskui
l"
535 POSITION N7, N9 : ? "Translation by A
lan J. Zett"
540 POSITION N10, 16 : ? "QUESTION ENTRY
MODULE"
  
```

Calculate free string space and define strings.

```

550 DIM PW$(N10), T$(45), TF$(N10), I$(LN
*N48), C$(N1), S$(N1), F$(8), QF$(12), AZ$(
N10), BEL$(4) : AZ=INT(FRE(NO)*0.3)
555 Q=INT(AZ/(LN*N48+152)) : H=Q : A=H*N3 :
DIM Q$(Q*(LN*N48)), H$(H*N38), A$(A*N38)
560 S$=CHR$(156) : BEL$=CHR$(253) : BEL$(N
2)=" "
570 T$(N1, 15)="TRUE OR FALSE " : T$(16,
30)="MULTIPLE CHOICE" : T$(31, 45)="FILL-
IN " : TF$(N1, N5)="FALSE"
580 TF$(6, N10)="TRUE " : S=53279
590 QN=NO : AP=NO
600 POSITION 6, 22 : ? "* PRESS ANY KEY T
O CONTINUE *" ; : POKE CRS, NO : GET #N1, L : 6
OSUB 700 : GOTO N1000
  
```

Set test data strings to spaces.

```

700 A$(N1)=" " : A$(A*N38)=" " : A$(N2)=A$
(N1) : B$=A$ : H$=A$ : RETURN
  
```

Main program loop.

```

1000 GRAPHICS NO : POKE CRS, N1 : POSITION
N2, N4 : RESTORE 20000 : GOSUB N11000
1110 IF IT<N4 OR IT>N5 OR QN=NO THEN 1
140
1120 ? BEL$ ; "ERASE TEST IN MEMORY. OK?"
; : GET #N1, A2 : IF A2<>N89 THEN GOTO N10
00
  
```

Erase old test.

```

1130 QN=NO : AP=NO : GOSUB 700
1140 IF (IT<N4 OR IT>N7) AND QN=NO THE
N ? BEL$ ; "NO TEST IN MEMORY" : FOR I=N1
  
```

```

TD 600:NEXT Z:GOTO N1000
1150 ON IT GOSUB 3000,5000,6000,3000,7
000,4000,8000,9900
1160 GOTO N1000

```

Get file name and test for existing file.

```

2000 GRAPHICS NO:?"FILE NAME FOR THIS
TEST:":? :? "=> ";:INPUT F#:GOSUB N15
*N1000:POKE CRS,N1
2010 ? :? "INSERT DISK AND PRESS A KEY
":GET #N1,L
2020 TRAP 2060
2030 OPEN #N2,N4,NO,QF#
2040 ? :? "QUIZ FILE '":F#;"' IS ALREA
DY":?"ON THE DISK - OK TO ERASE?":CLO
SE #N2
2045 GET #N1,L:IF L<>N89 THEN GOTO N2#
N1000
2050 GOTO 2070

```

Trap for "File Not Found" error.

```

2060 IF PEEK(195)<>170 THEN GOSUB 1010
0
2070 CLOSE #N2:RETURN

```

Check free question space. Cancel input if no more room.

```

3000 IF QN=Q OR AP=A THEN ? BEL#;"NO M
ORE ROOM FOR QUESTIONS";:FOR Z=N1 TO 6
00:NEXT Z:GOTO N1000

```

If not set, get test file name and question options.

```

3005 IF QN=NO THEN GOSUB N2*N1000
3010 IF ED=NO THEN GOSUB N4*N1000
3020 QV=N1

```

Input questions and type chosen.

```

3030 A1=AP+N1:GRAPHICS NO:?" QUESTION
# ";QN+N1:POKE CRS,N1
3040 POSITION N2,N2:T=TYPE:IF T=N4 THE
N ? "1=T/F 2=MULTI 3=FILL-IN 0=MENU
":? :? "TYPE?":GET #N1,T:T=N48
3045 IF T=NO OR T=N107 THEN RETURN
3050 IF T<NO OR T>N3 THEN 3040
3060 POSITION 21,NO:?" T$(T#N15-N14,T#N
15):H=N2:V=H:GOSUB N100*N100
3070 NQ=NO:GOSUB N100:POKE CRS,N1
3080 IF I$="" THEN 3620
3090 QN=QN+N1:Q$(QN#LN#N48-(LN#N48-N1
)),(QN#LN#N48)-N10)=I$

```

Input question responses.

```

3100 ? :? :IF T<>N2 THEN 3160
3110 ? " ANSWER CHOICES ":NC=N1
3120 ? NC;". ";:GOSUB N100
3130 IF I$="" THEN ? S#:NC=NC-N1:GOTO
3160
3140 AP=AP+N1:A$(AP#N38-N37,AP#N38)=I$

```

```

3150 ? :IF NC<N9 THEN NC=NC+N1:GOTO 31
20
3160 POKE CRS,NO:?" CORRECT RESPONSE
"
3170 ON T GOTO 3180,3210,3230

```

Input True/False response.

```

3180 POKE CRS,N1:?" T OR F ? ";:GET #N
1,X:?" CHR$(X):IF X<>N84 AND X<>70 THEN
3180
3190 TF=NO:IF X=N84 THEN TF=N1
3200 GOTO 3280
3210 POKE CRS,N1:?" S#;"NUMBER?";:GET #
N1,CC:CC=CC-N48:IF CC<N1 OR CC>NC THEN
3210
3220 ? :GOTO 3280
3230 NN=NO:GOSUB N100:?"
3240 NN=NN+N1:AP=AP+N1:A$(AP#N38-N37,A
P#N38)=I$:IF NN=N9 THEN 3270

```

Select an alternate response.

```

3250 ? " ALTERNATE ANSWER (RETURN IF N
ONE) ":GOSUB N100:?"
3260 IF I$<>"" THEN 3240
3270 ? S#;" CHARS TO MATCH (1-9, RETUR
N=ALL) ";:GET #N1,CM:CM=CM-N48:IF CM>N
9 AND CM<>107 THEN 3270
3275 ? :? :IF CM=N107 THEN CM=NO
3280 IF HYNT=NO THEN 3350

```

Input hint if applicable.

```

3290 ? " HINT (RETURN IF NONE) "
3300 GOSUB N100:?"
3310 H$(QN#N38-N37,QN#N38)=I$
3320 HS=N1:IF I$="" THEN HS=NO:GOTO 33
50
3330 IF PFLAG=NO THEN 3350

```

Input hint penalty and question value.

```

3340 ? S#;" HINT PENALTY, TENTHS (0-9)
";:GET #N1,HP:HP=INT(HP-47.5):IF HP<N
0 OR HP>N9 THEN 3340
3345 ? :?
3350 IF VFLAG=NO THEN 3500
3360 ? S#;" QUESTION VALUE ";:GET #N1,
QV:QV=QV-N48:IF QV<N1 OR QV>N9 THEN 33
60
3370 ?

```

Construct encoded question/response data.

```

3500 AZ$=STR$(T)
3510 AZ$(LEN(AZ$)+N1)=STR$(QV)
3520 IF T=N1 THEN AZ$(LEN(AZ$)+N1)=STR
$(TF):GOTO 3580
3530 I$="00":I$(N3)=STR$(A1):AZ$(LEN(A
Z$)+N1)=I$(LEN(I$)-N2,LEN(I$))
3540 X=NC:IF T=N3 THEN X=NN
3550 AZ$(LEN(AZ$)+N1)=STR$(X)

```

```

3560 X=CC:IF T=N3 THEN X=CM
3570 AZ$(LEN(AZ$)+N1)=STR$(X)
3580 AZ$(LEN(AZ$)+N1)=STR$(HS)
3590 AZ$(LEN(AZ$)+N1)=STR$(HP)
3600 Q$(QN#LN#N48-N9,(QN#LN#N48)-N1)=A
Z$
3610 GOTO 3030
3620 RETURN

```

Select question options.

```

4000 GRAPHICS NO:POKE CRS,N1:?" RESTOR
E 20010:EO=N1
4010 GOSUB N11000:TYPE=IT-N1:IF TYPE=N
0 THEN TYPE=N4
4020 GOSUB N11000:VFLAG=IT-N1
4030 GOSUB N11000:HYNT=N2-IT
4040 IF HYNT=NO THEN 4080
4050 GOSUB N11000:PFLAG=N1:IF IT=N2 TH
EN 4070
4060 PFLAG=NO:?" HINT PENALTY, TENTHS
(0-9)";:GET #1,HP:HP=INT(HP-47.5):IF
HP<NO OR HP>N9 THEN ? S#;:GOTO 4060
4070 POSITION N2,22:?" ALL OK?";:GET #
N1,L:IF L=78 THEN 4000
4080 IF L<>N89 THEN 4070
4090 RETURN

```

Review test question/response data.

```

5000 FOR I=N1 TO QN:GRAPHICS NO:I$=Q$(
I#LN#N48-(LN#N48-N1),(I#LN#N48)-N1):G0
SUB N9*N1000
5010 ? I;": ";T$(T#N15-N14,T#N15);"
VALUE=";V:?"
5020 ? I$:POKE CRS,N1
5030 IF T=N1 THEN ? "ANSWER: ";TF$(TF
+N1)#N5-N4,(TF+N1)#N5):? :GOTO 5120
5040 ? "ANSWERS: ";:IF T=N3 THEN 5080
5050 ? :? :? :FOR J=N1 TO N:JJ=A1+J-N1
:POSITION N2,PEEK(N84)-N1:?" CHR$(J+N48
+(128*(J=C)))";
5060 ? ". ";A$(JJ#N38-N37,JJ#N38):IF P
EEK(N84)>19 THEN ? "?";:GET #N1,L:?" CH
R$(156);
5070 NEXT J:GOTO 5120

```

Calculate portion of response to match.

```

5080 ? "(MUST MATCH ";:IF C=NO THEN ?
"ALL";:GOTO 5090
5085 ? C;
5090 ? " CHARACTERS)":? :? :FOR J=N1 T
O N:JJ=A1+J-N1:POSITION N2,PEEK(N84)-N
1
5100 ? A$(JJ#N38-N37,JJ#N38):IF PEEK(N
84)>19 THEN ? "?";:GET #N1,L:?" CHR$(15
6);
5110 NEXT J
5120 IF HS<>NO THEN ? "HINT (PENALTY="
;HP;"/10)":? :? :H$(I#N38-N37,I#N38)

```



TUNE IN NEXT MONTH FOR ANOTHER ASTEROID HANGER.

TRICKY TUTORIALS (tm)

There are many things that the ATARI computers can do either better, or easier than other small computers. The following series of programs is designed for anyone who is at least familiar with BASIC programming. What each tutorial offers is similar to an extensive magazine article with all discussion in as simple language as possible, plus you get MANY examples already typed in and running. The instruction manuals range from 10 to 50 pages, and some tutorials fill up a complete tape or disk. There is little overlap in what is taught, so anyone wanting to know all they can should buy them all (my banker thanks you). ATARI buys these from us to use in training their own people! Rave reviews have been published in ANTIC, ANALOG, CREATIVE COMPUTING, and even INFO WORLD. You trust INFO WORLD, don't you?

TT #1: DISPLAY LISTS—This program teaches you how to alter the program in the ATARI that controls the format of the screen. Normally, when you say "Graphics 8", the machine responds with a large Graphics 8 area at the top of the screen and a small text area at the bottom. Now, you will be able to mix various Graphics modes on the screen at the same time. The program does all of the difficult things (like counting scan lines). You will quickly be able to use the subroutines included in your own programs. **\$19.95**
16K Tape or 24K Disk.

TT #2: HORIZONTAL/VERTICAL SCROLLING—The information you put on the screen, either GRAPHICS or TEXT, can be moved up, down, sideways, or diagonally. We provide the basic methods and leave the rest up to your skill and imagination. Includes 18 examples to get you started, with several using a small machine language subroutine for smoothness. **\$19.95**
16K Tape or 24K Disk.

TT #3: PAGE FLIPPING—Now you don't have to redraw the screen every time you change the picture or text. You will learn how to have the computer draw the next screen you want to see while you are still looking at the previous screen, then flip to it instantly. You won't see it being drawn, so a complicated picture can seem to just appear. Depending on your memory size and which graphics or text modes you are using, you can instantly look at up to 50 pages. The basic method takes only 9 lines and the usefulness is infinite. **\$19.95**
16K Tape or 24K Disk.

TT #4: BASICS OF ANIMATION—This program shows you how to animate simple shapes (with some sound) using the PRINT and PLOT commands, and it also has a nice little PLAYER/MISSILE GRAPHICS game you can learn from. The P/M example is explained and will get you started on this complicated subject (more fully explained in TT #5). This would be an excellent way to start making your programs come alive on the screen with movement! Recommended for beginning users. **\$19.95**
16K Tape or 24K Disk.

TT #5: PLAYER/MISSILE GRAPHICS—Learn to write your own games and other animated applications! The tutorial begins with many small examples that compliment the 50 page manual, then gradually builds up to a complete game where everything you need to know is fully explained. Also included are two machine language utilities that you can use to animate Players with from BASIC. Next we include two of the best editors currently available; one for editing playfield shapes (backgrounds); and one to edit your players, and all in glorious Technicolor! Everything except the two editors run in 16K Tape or 32K Disk. **\$29.95**

TT #6: SOUND AND MUSIC—Unless you have spent many years experimenting with the four voice channels, you will learn a lot from this one! Learn to play standard notes, chords, and whole songs using some simple "tricks". One of the nicest parts are the examples of special sound effects that you can refer to whenever you need a sound for a program or to impress a friend. This program will be of interest to all ages and levels of experience! **\$19.95**
16K Tape or 24K Disk.

SPECIAL DISCOUNT

Order the first six tutorials in a 3-ring binder for \$99.95, a \$30.00 savings!

TT #7: DOS UTILITIES—We at Educational Software have been shocked by some of the prices others are charging to offer you small utilities to help in the use of your Disk Drive. We now offer you all of the following plus explanation as to how each was written, and how to use them: A UNIQUE MENU PROGRAM, AN AUTORUN SYS BUILDER, DISK INSPECTOR (LOOK AT SECTORS), DISK JACKET PRINTER, AUTOMATIC FORMATTER, RECORD SAVE AND LOAD UTILITY. **\$29.95**
32K Disk Only.

MASTER MEMORY MAP (tm)

This book is the most valuable source of information for your ATARI you can buy. It starts out by explaining how to PEEK and POKE values into memory, so that even new computer owners can use many of these "Tricks". Then you are given 32 pages of the memory locations that are the most useful, along with hints on how to use many of the locations. Finally, it includes hints on problems you may be having with the computer and discusses the new Graphics modes 9 to 11. Even ATARI buys this book from us! **\$6.95**

CONTACT YOUR LOCAL DEALER or ORDER BELOW

We have other fine programs for ATARI computers. Write for a catalog. Send us your programs to sell too!

USER SUBMITTED PROGRAMS

MINI-DATABASE/DIALER—stores and edits up to 8 lines of information such as names & addresses, phone numbers, messages, inventories, or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature: If your file includes phone numbers and your phone company allows touch-tone phone signals, the program will DIAL THE PHONE NUMBER FOR YOU! **\$24.95**
16K Tape or 24K Disk.

THE GRAPHICS MACHINE—allows the ATARI to act like more expensive graphics computers using simple commands like line, box, circle, polygon, fill, and savescreen to get a high resolution picture you can save on disk in only five seconds! Many more features! **\$19.95**
48K Disk Only.

BOB'S BUSINESS—14 small business type programs accessed from a common menu. **\$14.95**
16K Tape or 32K disk.

MINI WORDPROCESSOR—A simple text editor to write, save, and print several pages at a time. **\$19.95**
32K Tape or Disk.

KID'S #1—Includes a MATH QUIZ, a children's TREASURE HUNT, and a DIALOGUE program. **\$14.95**
16K Tape or 24K Disk. 3 for . . .

KID'S #2—SPELLING BEE, WORD SCRAMBLE, and TOUCH. **\$14.95**
16K Tape or 24K Disk. 3 Educational games for . . .

PLAYER PIANO—Turns your keyboard into a mini-piano and more. **\$14.95**
24K Tape or 32K Disk.

GRAPHIC SYMBOL LABELS—for your keyboard to remind you of the built-in Graphics symbols. 2 complete sets for . . . **\$2.95**

OUR NEWEST PROGRAMS

DOG DAZE—Two cute little doggies race for the fire hydrants, shoot their bones, and just have a lot of fun! A fast action program for all ages. **\$16.95**
8K Tape or 16K Disk, in machine language.

* OUR BEST GAME *

SPACE GAMES—Our family is being attacked by ALIENS, and only you can save us. A comic book manual will guide you through three games that test your ability in space skills. Includes ALIENS, SURVIVE, and ROBOT ATTACK, and is for all ages. The first two games require 16K for Tape. The last game and all Disk users need 32K. **\$24.95**

MATHS FOR FUN—Another ENGLISH import teaching basic math skills. Very colorful and enjoyable to use. For ages 5 to 16. **\$19.95**
16K Tape or 24K Disk.

MARATHON—This is a unique math quiz for one or two players. You are in a race to move your runner across the screen first! There are four levels of play with five modes of operation for each. The game uses joysticks for all input, so play is easy for young children. This wonderful learning tool is imported from ENGLAND for your learning pleasure. Your kids will never even notice they are playing an EDUCATIONAL program. **\$19.95**
16K Tape or 24K Disk.

GRADE BOOK—This nice record keeper will maintain a file of 35 students' test scores along with comments. Up to 7 subjects are allowed. **\$24.95**
24K Tape or 32K Disk. A printer is optional.

To order COD, VISA or MasterCard call — (408) 476-4901

By mail include \$3.00 postage (USA only) or \$1.50 Memory Map only—
California residents add 6.5% TAX.—Specify Tape or disk.



EDUCATIONAL SOFTWARE inc.

5425 JIGGER DR.
SOQUEL, CA 95073

```
5130 POKE CRS,N1:? "PRESS A KEY TO C
ONTINUE ";:GET #N1,L
5140 NEXT I
5150 RETURN
```

Save test file data.

```
6000 GRAPHICS NO:POKE CRS,N1:POSITION
N2,N5:? "USE FILE NAME ";F$;"?:GET
#N1,L:IF L<N89 THEN GOSUB 2000
```

Check password.

```
6020 IF PW$="" THEN ? :? "USE PASSWORD
";PW$;"?:GET #N1,L:IF L=N89 THEN 6
040
```

```
6030 POKE CRS,NO:? :? "PASSWORD FOR TH
IS TEST";:INPUT PW$
```

Encode password.

```
6040 I$="" :AZ$="00":AZ$(N3)=STR$(QN):I
$(LEN(I$)+N1)=AZ$(LEN(AZ$)-N2,LEN(AZ$)
)
```

```
6045 AZ$="00":AZ$(N3)=STR$(AP):I$(LEN(
I$)+N1)=AZ$(LEN(AZ$)-N2,LEN(AZ$)):I$(L
EN(I$)+N1)=PW$
```

```
6050 POKE CRS,N1:? :? "SAVING ";F$;"
TO DISK"
```

```
6060 TRAP 6140
```

Print data to disk.

```
6070 OPEN #N2,8,NO,QF$
6110 ? #N2;LN:? #N2;I$:FOR I=N1 TO QN:
? #N2;Q$(I#LN#N48-(LN#N48-N1),(I#LN#N4
8)-N1):NEXT I
```

```
6120 FOR I=N1 TO AP:? #N2;A$(I#N38-N37
,I#N38):NEXT I
```

```
6130 FOR I=N1 TO QN:? #N2;H$(I#N38-N37
,I#N38):NEXT I:GOTO 6150
```

```
6140 GOSUB 10100
```

```
6150 CLOSE #N2:RETURN
```

Load in old test file data.

```
7000 GRAPHICS NO:? :GOSUB 700
```

```
7010 TRAP 7090
```

```
7020 ? "INPUT FILE NAME";:INPUT F$:GOS
UB 15000:POKE CRS,N1:OPEN #N2,N4,NO,QF
$
```

Decode password.

```
7050 INPUT #N2;LN:INPUT #N2;I$:QN=VAL(
I$(N1,N3)):AP=VAL(I$(N4,6)):PW$=I$(N7)
```

Read test file data.

```
7060 FOR I=N1 TO QN:INPUT #N2;I$:Q$(I#
LN#N48-(LN#N48-N1),(I#LN#N48)-N1)=I$:N
EXT I
```

```
7070 FOR I=N1 TO AP:INPUT #N2;I$:A$(I#
N38-N37,I#N38)=I$:NEXT I
```

```
7080 FOR I=N1 TO QN:INPUT #N2;I$:H$(I#
N38-N37,I#N38)=I$:NEXT I:GOTO 7100
```

```
7090 IF PEEK(195)=170 THEN ? :? "FILE
```

```
NOT FOUND";BEL$;FOR Z=N1 TO 300:NEXT Z
:GOTO 7100
```

```
7095 GOSUB 10100
```

```
7100 CLOSE #N2:RETURN
```

Print out test.

```
8000 POKE CRS,N1:POSITION N2,20:? "PLE
ASE TURN ON YOUR PRINTER"? "THEN PRES
S ANY KEY";:GET #N1,L:TRAP 8070
```

```
8010 OPEN #N2,8,NO,"P":FOR I=N1 TO QN
:I$=Q$(I#LN#N48-(LN#N48-N1),(I#LN#N48)
-N1)
```

```
8020 GOSUB N9#N1000:? #N2;I$. ";I$:?
#N2
```

```
8030 IF T=N1 THEN ? #N2;" ___ TRUE":? #
N2;" ___ FALSE":GOTO 8060
```

```
8040 IF T=N3 THEN ? #N2;"-----
___":GOTO 8060
```

```
8050 FOR J=N1 TO N:L=A1+J-N1:? #N2;" _
_";J". ";A$(L#N38-N37,L#N38):NEXT J
```

```
8060 ? #N2:? #N2:? #N2:NEXT I:CLOSE #N
2:RETURN
```

```
8070 TRAP N4#N1000:GOTO N1000
```

Separate encoded question/response data.

```
9000 L=LN#N48-N10:T=VAL(I$(L+N1,L+N1))
:V=VAL(I$(L+N2,L+N2))
```

```
9010 IF T=N1 THEN TF=VAL(I$(L+N3,L+N3)
):GOTO 9030
```

```
9020 A1=VAL(I$(L+N3,L+N5)):N=VAL(I$(L+
6,L+6)):C=VAL(I$(L+N7,L+N7))
```

```
9030 X=N9-N4*(T=N1):HS=VAL(I$(L+X-N1,L
+X-N1)):HP=VAL(I$(L+X,L+X))
```

```
9040 I$=I$((LN#N48-(LN#N48-N1)),(LN#N4
8)-N10)
```

```
9050 RETURN
```

Exit from program.

```
9900 POKE CRS,N1:POSITION N2,20:? "DO
YOU WANT TO QUIT?";:GET #N1,L:IF L=N89
THEN END
```

```
9910 RETURN
```

Clear to end of screen routine.

```
10000 POKE CRS,N1:COLOR 32:PLOT H,V:DR
AWTO N39,V:FOR L=V+N1 TO 23:PLOT NO,L:
DRAWTO N39,L:NEXT L:POSITION H,V:RETUR
N
```

Error trap/display routine.

```
10100 TRAP N4#N1000:POKE CRS,N1:? :?
" ERROR ";PEEK(195);BEL$:? " IN LIN
E ";PEEK(186)+256*PEEK(187);BEL$
10110 FOR L=N1 TO N1000:NEXT L:RETURN
```

Read menu heading and options.
Print heading.

```
11000 READ I$,N:V=PEEK(N84)? " ";I$
```

Print options.

```
11010 FOR I=N1 TO N
11020 READ I$
11030 ? " ";I$. ";I$
11040 POKE S,NO:POKE S,N1:POKE S,N2
11050 NEXT I
```

Display menu prompt.

```
11060 POSITION N2,V:? "=>";
11070 GET #N1,IT:IT=IT-N48
11080 POKE S,NO:POKE S,N1:POKE S,N2
11090 IF IT<N1 OR IT>N THEN 11060
11100 POSITION N2,V:? " ";VV=V+IT:PO
SITION N2,VV:? "=>";
```

```
11110 GET #N1,X
```

```
11120 IF X<>155 THEN IT=X-N48:POSITION
N2,VV:? " ";:GOTO 11080
```

```
11130 POSITION N2,V+N+N2:RETURN
```

File name manipulator.

```
15000 QF$="D":QF$(N3,N10)=F$:QF$(LEN(
QF$)+N1)=".Q":RETURN
```

Data for question options.

```
20000 DATA QUESTION ENTRY MAIN OPTION
MENU,8,ADD TO TEST IN MEMORY,REVIEW
TEST IN MEMORY,SAVE TEST TO DISK
```

```
20005 DATA BEGIN A NEW TEST,LOAD A TES
T FROM DISK,RESET QUESTION ENTRY OPTIO
NS,PRINT TEST ON PRINTER,QUIT
```

```
20010 DATA QUESTION TYPES,4,MIXED TY
PES,TRUE/FALSE,MULTIPLE CHOICE,FILL-IN
20020 DATA QUESTION VALUES,2,ALL THE
SAME,VARIABLE
```

```
20030 DATA HINT OPTION,2,ON,OFF
```

```
20040 DATA HINT PENALTY,2,FIXED PERC
ENTAGE,VARIABLE PERCENTAGE
```

ATARI® SWAT TABLE FOR: CATS DISK VERSION

LINES	SWAT CODE	LENGTH
1 - 200	BB	373
210 - 550	SE	593
555 - 1120	KE	547
1130 - 2060	KW	448
2070 - 3080	PU	371
3090 - 3200	AE	363
3210 - 3310	XR	390
3320 - 3540	JH	362
3550 - 4030	LB	237
4040 - 5050	KF	481
5060 - 6000	TR	446
6020 - 6150	QW	460
7000 - 8010	YD	545
8020 - 9050	ZG	472
9900 - 11060	SJ	363
11070 - 20020	BH	536
20030 - 20040	SS	87

```

#####
$   ATARI BASIC   $
$ 'CATS: TAPE VERSION' $
$ Author: Alan J. Zett $
$ (c) 1982, SoftSide $
#####

```

```

3005 REM
6000 GRAPHICS NO:POKE CRS,N1:POSITION
N2,N3
6020 IF PW$="" THEN ? :? "USE PASSWORD
";PW$;"?":GET #N1,L:IF L=N89 THEN 6
040
6030 POKE CRS,NO:?:? "PASSWORD FOR TH
IS TEST";:INPUT PW$
6040 I$="" :AZ$="00":AZ$(N3)=STR$(QN):I
$(LEN(I$)+N1)=AZ$(LEN(AZ$)-N2,LEN(AZ$)
)
6045 AZ$="00":AZ$(N3)=STR$(AP):I$(LEN(
I$)+N1)=AZ$(LEN(AZ$)-N2,LEN(AZ$)):I$(L
EN(I$)+N1)=PW$
6050 POKE CRS,N1:?:? "POSITION TAPE,
PRESS RETURN TO SAVE":?
6060 TRAP 6140
6070 OPEN #N2,B,NO,"C":?: "SAVING ";
6110 ? #N2;LN:?: #N2;I$:FOR I=N1 TO QN:
60SUB 7095:?: #N2;Q$(I#LN#N48-(LN#N48-N
1),(I#LN#N48)-N1):NEXT I
6120 FOR I=N1 TO AP:60SUB 7095:?: #N2;A
$(I#N38-N37,I#N38):NEXT I
6130 FOR I=N1 TO QN:60SUB 7095:?: #N2;H
$(I#N38-N37,I#N38):NEXT I:60TO 6150
6140 60SUB 10100
6150 CLOSE #N2:RETURN
7000 GRAPHICS NO:?: :60SUB 700
7010 TRAP 7090
7020 ? "POSITION TAPE, PRESS RETURN TO
LOAD":OPEN #N2,N4,NO,"C":?:? "LOADI
NG ";
7050 INPUT #N2;LN:INPUT #N2;I$:QN=VAL(
I$(N1,N3)):AP=VAL(I$(N4,6)):PW$=I$(N7)
7060 FOR I=N1 TO QN:60SUB 7095:INPUT #
N2;I$:Q$(I#LN#N48-(LN#N48-N1),(I#LN#N4
8)-N1)=I$:NEXT I
7070 FOR I=N1 TO AP:60SUB 7095:INPUT #
N2;I$:A$(I#N38-N37,I#N38)=I$:NEXT I
7080 FOR I=N1 TO QN:60SUB 7095:INPUT #
N2;I$:H$(I#N38-N37,I#N38)=I$:NEXT I:60
TO 7100
7090 60SUB 10100:60TO 7100
7095 POSITION PEEK(85)-1,PEEK(N84):? C
HR$(32+128*(INT(I/2)*2=I));:RETURN
7100 CLOSE #N2:RETURN
20000 DATA QUESTION ENTRY MAIN OPTION
MENU ,8,ADD TO TEST IN MEMORY,REVIEW
TEST IN MEMORY,SAVE TEST TO TAPE
20005 DATA BEGIN A NEW TEST,LOAD A TES
T FROM TAPE,RESET QUESTION ENTRY OPTIO
NS,PRINT TEST ON PRINTER,QUIT

```

ATARI® SWAT TABLE FOR:
CATS TAPE VERSION

LINES	SWAT CODE	LENGTH
1 - 200	CY	373
210 - 550	SY	593
555 - 1120	OZ	547
1130 - 3050	IP	434
3060 - 3170	RR	352
3180 - 3280	ZS	425
3290 - 3510	VO	315
3520 - 4000	EU	288
4010 - 5020	RI	383
5030 - 5130	OC	519
5140 - 6120	KZ	501
6130 - 7095	RD	521
7100 - 9020	YB	544
9030 - 11030	CH	413
11040 - 20000	VR	363
20005 - 20040	IJ	305



APPLE™

```

#####
$   APPLESOFT BASIC   $
$ 'CATS' - DISK VERSION $
$ AUTHOR: JON R. VOSKUIL $
$ (C) 1982   SOFTSIDE $
#####

```

10 GOTO 500

Custom line input routine.

```

100 I$ = "":V = PEEK (37) + 1:H =
PEEK (36) + 1
110 GET C$:C = ASC (C$): IF C <
32 THEN 150

```

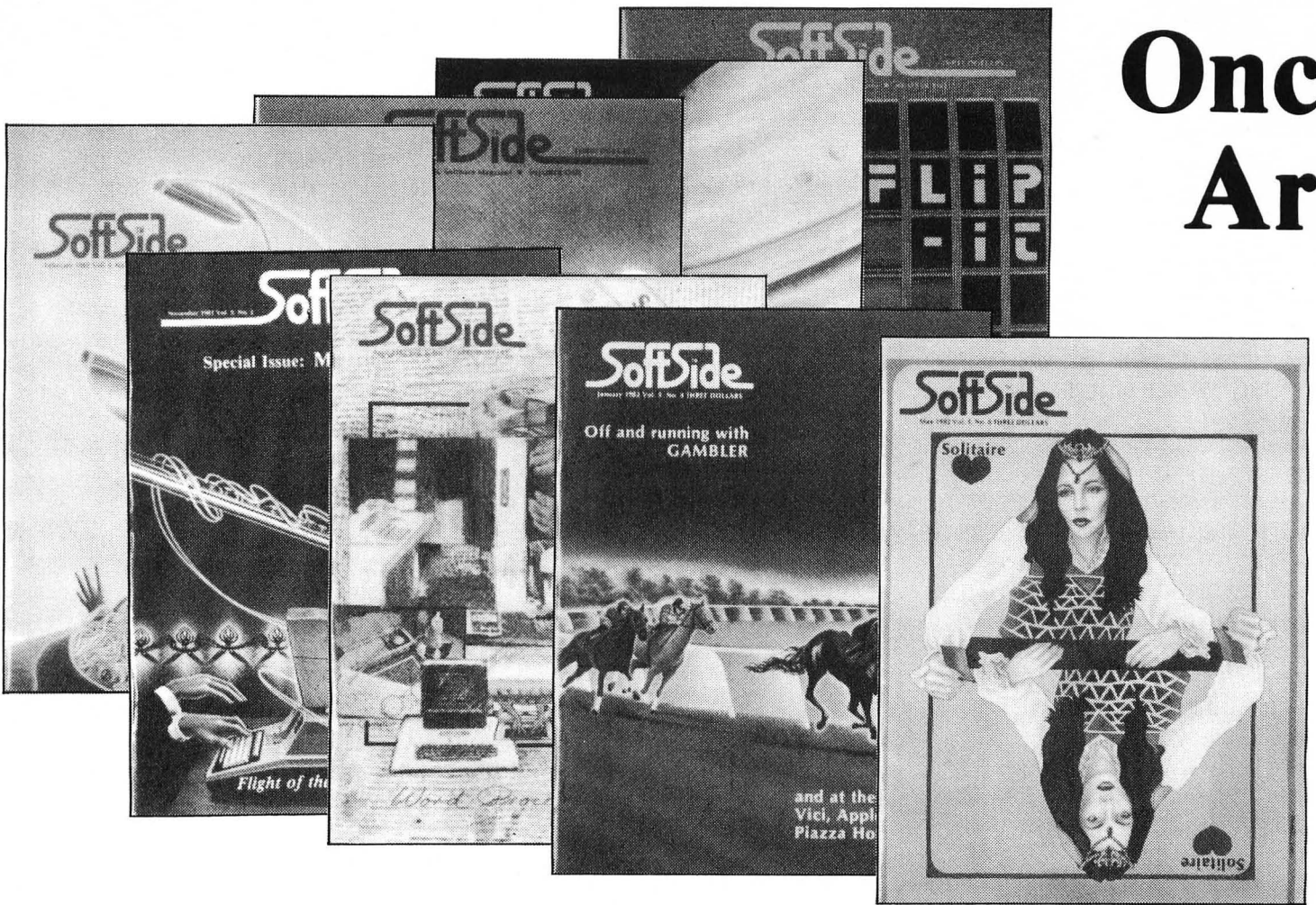
```

120 IF C = 34 THEN C = 39:C$ = "
"
130 IF LEN (I$) = 230 THEN PRINT
BEL$;: GOTO 110
140 I$ = I$ + C$: PRINT C$;: GOTO
110
Process control commands.
150 IF C = 13 THEN RETURN
160 IF C = 21 THEN C = 95:C$ = CHR$
(C): GOTO 130
170 IF C = 5 THEN VTAB V: HTAB
H: PRINT SPC( 255): VTAB V:
HTAB H: GOTO 100
180 IF C < > 8 THEN 110
190 L = LEN (I$): IF L < 2 THEN
I$ = "": GOTO 210
200 I$ = LEFT$ (I$,L - 1)
210 IF L > 0 THEN PRINT C$;" ";
C$;
220 GOTO 110
Display title page and initialize
program.
500 TEXT : HOME : VTAB 8
510 PRINT TAB( 17)"C A T S"
520 VTAB 12: PRINT TAB( 12)"BY
JON R. VOSKUIL"
530 VTAB 18: INVERSE : HTAB 10: PRINT
"QUESTION ENTRY MODULE": NORMAL
540 D$ = CHR$ (4): PRINT D$"NOMD
NC,I,0"
550 DIM B$(100),A$(300),H$(100),
T$(3),N$(9),TF$(1)
560 S$ = "
";U$ = CHR$
(95):U3$ = U$ + U$ + U$:QUO$
= CHR$ (34)
570 T$(1) = "TRUE OR FALSE":T$(2)
= "MULTIPLE CHOICE":T$(3) =
"FILL-IN":TF$(0) = "FALSE":T
F$(1) = "TRUE"
580 KEY = - 16384:PRESS = 128:CL
R = - 16368:BR$ = CHR$ (93
) + CHR$ (91):S = - 16336:
BEL$ = CHR$ (7):B$ = CHR$
(8)
590 QN = 0:AP = 0: 60SUB 15000
600 VTAB 24: HTAB 5: PRINT BR$;"
PRESS ANY KEY TO CONTINUE "
;BR$;: 60SUB 10000
Main program loop.
1000 HOME : VTAB 5:N$ = " QUESTI
ON ENTRY MAIN OPTIONS MENU "
:N = 8
1010 N$(1) = "ADD TO TEST IN MEMO
RY"
1020 N$(2) = "REVIEW TEST IN MEMO
RY"

```

continued on page 32

Once Are



Back Issues

from

SoftSide™

August 1980

"You Can Have Sound" — Apple
"Rom the Robot" — Apple
"Masters Golf" — ATARI®
"Grand Prix" — TRS-80®

September 1980

"Goal" — 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®

They're Gone... They Gone Forever?

If you like the programs, reviews, and programming information in this issue of *SoftSide*...think of what's waiting for you in past issues!

Exciting games like *Defense...Mini-Golf...Micro Man...*
Great Graphics like *Old Glory...Titan...Flight of the Bumblebee...*
PLUS...databases, utility programs, educational programs, and more!

It's all here in the Back Issues of *SoftSide* magazine. And many of these issues are still available for your enjoyment. But not all. Several issues are SOLD OUT...others are available but supply is limited.

So check out the issues and features listed below and order today!

- Back Issues \$3.50 ea.*
- Back Issue on cassette \$9.95 ea.*
- Back Issue on disk \$14.95 ea.*
- Back Issue w/ Enhanced Disk Version
(contains an additional program) \$19.95 ea.*

Use our bind-in order card provided elsewhere in this issue to order your Back Issue copies, cassettes, disk and enhanced disk versions of *SoftSide*.

For magazine orders only (minimum 3 issues) send order card and payment to:
SoftSide Magazine 515 Abbott Drive Broomall, PA 19088

For magazine plus cassette or disk orders...and enhanced disk orders...send order card and payment to: **SoftSide Magazine, 6 South St., Milford, NH 03055**

*Prices good for USA orders only — for foreign pricing see page 16 for information.



January 1982

- "Gambler" — All Systems
- "Microtext 1.1" — All Systems
- "Apple Capture" — Apple
- "Piazza Hotel" — ATARI®
- "TRS-Man" — TRS-80®

Enhanced Disk Versions

- "Nuclear Submarine Adventure" — Apple, TRS-80®
- "Death Star" — ATARI®

February 1982

- "Space Rescue" — All Systems
- "Rubicube" — Apple
- "Defense" — ATARI®
- "Maze Sweep" — TRS-80®

Enhanced Disk Versions

- "Andorra" — Apple
- "Kismet II" — ATARI®
- "Help Package" — TRS-80®

March 1982

- "Hexapawn" — All Systems
- "Magical Shape Machine" — Apple
- "Outer Space Attack" — ATARI®
- "Killer Cars" — TRS-80®

Enhanced Disk Version

- "PEEKER/POKER" — Apple
- "Curse of the Pharaoh" — ATARI®
- "Warpath" — TRS-80®

April 1982

- "Microtext" — All Systems
- "Poster Maker" — Apple
- "ATARI® Banner Machine" — ATARI®
- "Database" — TRS-80®

Enhanced Disk Versions

- "Semaphore" — Apple
- "Renumbering for the ATARI®" — ATARI®
- "Screen Print" — TRS-80®

May 1982

- "Solitaire" — All Systems
- "Micro-Man" — Apple
- "Cross Reference" — ATARI®
- "Ladders" — TRS-80®

Enhanced Disk Versions

- "List Formatter" — Apple
- "Robot Battle" — ATARI®
- "Breakthru" — TRS-80®

#30

- "Escape from the Dungeons of the Gods" — All Systems
- "SWAT" — All Systems
- "Code Breaker" — Apple
- "Sabotage" — ATARI®
- "Piazza Hotel" — TRS-80®

January 1981

- "Developing Data Base V" — All Systems
- "Convoy" — Apple and TRS-80®
- "Angle Cannon" — ATARI®
- "Ship Destroyer" — TRS-80®

February 1981

- "Developing Data Base VI" — All Systems
- "Miner" — All Systems
- "Mini-Golf" — ATARI® and TRS-80®
- "Long Distance" — TRS-80®

March 1981

- "Developing Data Base VII" — All Systems
- "Strategy Strike" — Apple and TRS-80®
- "Flags" — ATARI®
- "Volcano" — TRS-80®

April 1981

- "Battle At Sea" — Apple
- "Convoy" — ATARI®
- "Dominoes" — TRS-80®

May 1981

- "Galaxia" — Apple
- "Dodge" — ATARI®
- "Orienteering At Jacques Coulee" — TRS-80®

June 1981

- "Old Glory" — All Systems
- "Word-Search Puzzle Generator" — All Systems
- "Anallist" — TRS-80®

July 1981

- "Chemistry Drill" — All Systems
- "Volleyball" — Apple
- "Space Lander" — ATARI®
- "Word Wars" — TRS-80®

August 1981

- "Quest 1" — All Systems
- "Shark" — Apple
- "Dairy Farming" — ATARI®
- "Compu-Sketch" — TRS-80®

September 1981

- "Flip-It" — All Systems
- "Orienteering at Jacques Coulee" — Apple
- "Exterminate" — TRS-80®

Enhanced Disk Version

- "NEWBASIC" — TRS-80® Mod-I

October 1981

- "Leyte" — All Systems
- "Developing Data Base" — Apple
- "Character Generator" — ATARI®
- "Envyrn™" — TRS-80®

Enhanced Disk Versions

- "Super Dairy Farming" — Apple
- "Gameplay" — TRS-80®

November 1981

- "Flight of the Bumblebee" — All Systems
- "Music Machine" — Apple
- "Music Programmer" — ATARI®
- "Music Editor" — TRS-80®

Enhanced Disk Versions

- "National Anthems" — Apple
- "Volleyball" — ATARI®
- "Mean Checkers Machine" — TRS-80®

December 1981

- "Titan" — All Systems
- "Aircraft Commander" — Apple
- "Developing Data Base" — ATARI®
- "Electronics Assistant" — TRS-80®

Enhanced Disk Versions

- "Bobsledding" — Apple
- "Survive" — ATARI®
- "Konane" — TRS-80®

```

1030 N$(3) = "SAVE TEST TO DISK"
1040 N$(4) = "BEGIN A NEW TEST"
1050 N$(5) = "LOAD A TEST FROM DI
SK"
1060 N$(6) = "RESET QUESTION ENTR
Y OPTIONS"
1070 N$(7) = "PRINT TEST ON PRINT
ER"
1080 N$(8) = "QUIT"
1100 GOSUB 11010
1110 IF IT < 4 OR IT > 5 OR QN =
0 THEN 1140
1120 PRINT " ERASES TEST IN ME
MORY. OK? "; GOSUB 10000: IF
X$ < > "Y" THEN 1000

```

Erase old test.

```

1130 QN = 0: AP = 0: PRINT : PRINT
" ERASING TEST ... "; FOR
X = 1 TO 100: Q$(X) = "": A$(X
) = "": A$(X + 100) = "": A$(X
+ 200) = "": H$(X) = "": NEXT
1140 IF (IT < 4 OR IT = 7) AND Q
N = 0 THEN PRINT " NO TES
T IN MEMORY": FOR Z = 1 TO 1
000: NEXT Z: GOTO 1000
1150 QN IT GOSUB 3000,5000,6000,
3000,7000,4000,8000,9900
1160 GOTO 1000

```

Get file name and check for existing file.

```

2000 HOME : PRINT "FILE NAME FOR
THIS TEST: "; PRINT : INPUT
"=> "; F$: QF$ = "Q." + F$
2010 PRINT : PRINT "INSERT INITI
ALIZED DISK AND PRESS A KEY"
: GOSUB 10000
2020 ONERR GOTO 2060
2030 PRINT : PRINT D$"VERIFY"QF$
2040 PRINT : PRINT "QUIZ FILE '"
F$"' : PRINT "IS ALREADY ON
THIS DISK. OK TO ERASE? "; GOSUB
10000: IF X$ < > "Y" THEN 2
000
2050 PRINT : PRINT D$"UNLOCK"QF$
: PRINT D$"DELETE"QF$: GOTO
2080

```

Trap for "File Not Found" error.

```

2060 E = PEEK (222): CALL FIX: IF
E = 6 THEN 2080
2070 PRINT : PRINT BEL$; "ERROR:
CODE "E: GOSUB 10000: GOTO 2
000
2080 POKE 216,0: RETURN

```

Check free space. Cancel input if none left.

```

3000 HOME : PRINT "CHECKING SPAC

```

```

E ... "; IF FRE (0) < 800 THEN
PRINT : INVERSE : PRINT BR$
; " NO MORE ROOM FOR QUESTION
S "; BR$: NORMAL : FOR Z = 1 TO
1000: NEXT : GOTO 1000

```

If not set, get test file name and question options.

```

3005 IF QN = 0 THEN GOSUB 2000
3010 IF NOT ED THEN GOSUB 4000
3020 QV = 1

```

Select questions type and input question.

```

3030 A1 = AP + 1: HOME : INVERSE
: PRINT " QUESTION # "; QN +
1; ": ": NORMAL
3040 VTAB 3: HTAB 1: X$ = "": T =
TYPE: IF T = 4 THEN PRINT "
TYPE? 1=T/F 2=MULTI 3=FIL
L-IN 0=MENU": GOSUB 10000:
T = VAL (X$)
3045 IF X$ = "0" OR X$ = CHR$ (
13) THEN RETURN
3050 IF T < 1 OR T > 3 THEN 3040
3060 VTAB 1: HTAB 20: PRINT T$(T
): CALL - 958: PRINT
3070 GOSUB 100
3080 IF I$ = "" THEN 3620
3090 QN = QN + 1: Q$(QN) = I$

```

Input question responses.

```

3100 PRINT : PRINT : IF T < > 2
THEN 3160
3110 INVERSE : PRINT " ANSWER CH
OICES: "; NORMAL : NC = 1
3120 PRINT NC. " "; GOSUB 100
3130 IF I$ = "" THEN HTAB 1: NC =
NC - 1: GOTO 3160
3140 AP = AP + 1: A$(AP) = I$
3150 PRINT : IF NC < 9 THEN NC =
NC + 1: GOTO 3120
3160 INVERSE : PRINT " CORRECT R
ESPONSE: "; NORMAL
3170 QN T GOTO 3180,3210,3230
3180 INPUT "T OR F? "; X$: X$ = LEFT$
(X$,1): IF X$ < > "T" AND X
$ < > "F" THEN 3180
3190 TF = 0: IF X$ = "T" THEN TF =
1
3200 GOTO 3280
3210 INPUT "NUMBER? "; X$: CC = VAL
(X$): IF CC < 1 OR CC > NC THEN
3210
3220 GOTO 3280
3230 NN = 0: GOSUB 100: PRINT
3240 NN = NN + 1: AP = AP + 1: A$(A
P) = I$: IF NN = 9 THEN 3270
3250 INVERSE : PRINT " ALTERNATE
ANSWER (RETURN IF NONE): ";
NORMAL : GOSUB 100: PRINT

```

```

3260 IF I$ < > "" THEN 3240
3270 INVERSE : INPUT " CHARS TO
MATCH (1-9, RETURN=ALL): "; X
$: CM = VAL (X$): IF CM > 9 THEN
CM = 9

```

```

3280 IF NOT HYNT THEN 3350

```

Input hint, if applicable.

```

3290 INVERSE : PRINT " HINT (RET
URN IF NONE): "; NORMAL
3300 GOSUB 100: PRINT
3310 H$(QN) = I$
3320 HS = 1: IF I$ = "" THEN HS =
0: GOTO 3350
3330 IF NOT PFLAG THEN 3350

```

Input hint penalty and question value.

```

3340 INVERSE : INPUT " HINT PENA
LTY, TENTHS (0-9): "; X$: NORMAL
: HP = INT ( VAL (X$) + .5):
IF HP < 0 OR HP > 9 THEN 33
40
3350 IF NOT VFLAG THEN 3500
3360 INVERSE : INPUT " QUESTION
VALUE: "; X$: NORMAL : QV = VAL
(X$): IF QV < 1 OR QV > 9 THEN
3360

```

Construct encoded question/response data.

```

3500 Q$ = STR$ (T)
3510 Q$ = Q$ + STR$ (QV)
3520 IF T = 1 THEN Q$ = Q$ + STR$
(TF): GOTO 3580
3530 X$ = "00" + STR$ (A1): X$ =
RIGHT$ (X$,3): Q$ = Q$ + X$
3540 X = NC: IF T = 3 THEN X = NN
3550 Q$ = Q$ + STR$ (X)
3560 X = CC: IF T = 3 THEN X = CM
3570 Q$ = Q$ + STR$ (X)
3580 Q$ = Q$ + STR$ (HS)
3590 Q$ = Q$ + STR$ (HP)
3600 Q$(QN) = Q$ + Q$(QN)
3610 GOTO 3030
3620 RETURN

```

Select question options.

```

4000 HOME : PRINT : RESTORE : ED =
1
4010 GOSUB 11000: TYPE = IT - 1: IF
TYPE = 0 THEN TYPE = 4
4020 GOSUB 11000: VFLAG = IT - 1
4030 GOSUB 11000: HYNT = 2 - IT
4040 IF NOT HYNT THEN 4080
4050 GOSUB 11000: PFLAG = 1: IF I
T = 2 THEN 4070
4060 PFLAG = 0: INPUT "HINT PENAL
TY, TENTHS (0-9): "; HP: HP =
INT (HP + .5): IF HP > 9 OR
HP < 0 THEN 4060

```



The Book

That's an Open Door to the Computer Future

KIDS & THE APPLE is its name, and its game is to prepare your child, or any child, to take his or her place as a member of the computer generation by teaching them the mysteries of the Apple* computer in ways they'll love and enjoy. Don't be surprised if you will also learn along with your child.

The kids of today are fascinated by computers to start with. And that's great, because it means **they're eager** to learn. But, until this book by Edward H. Carlson, learning about the Apple was a fumbling, stumbling effort for a child.

KIDS & THE APPLE was designed in every aspect to lead them gently, interestingly yet quickly into the computer world. First, it's a large 8½ by 11 book which can be opened flat for ease of use. Second, there are 35 chapters, each one building upon the knowledge of the prior chapter – and it's loaded with dozens and dozens of cartoons which make a point as they amuse.



Then, there are special sections for a parent or teacher to use so they can work along with the kids, if they wish, and help them over any rough spots.

Perhaps the major reasons the kids will love this book is that it is **truly** written so they can easily understand it (without a lot of confusing technical language) . . . and that they see on-screen-results almost immediately! Right away they realize they'll soon be programming their Apple, making their own games! . . . or creating other programs for school or work or to play.

The computer world is roaring toward us. To be successful at work, school or even play, a child will have to be knowledgeable about computers. Make sure your favorite child is prepared for the challenge. With KIDS & THE APPLE at his side, he'll enjoy learning and you'll know you've prepared him or her for a successful future. Only \$19.95.

At computer stores, or from:

The educational/book division of
 **DATAMOST**TM

9748 Cozycroft Ave.
 Chatsworth, CA 91311 (213) 709-1202

*Apple is a trademark of Apple Computer, Inc.

```

4070 VTAB 23: HTAB 1: PRINT "ALL
      OK? "; GET X$: PRINT X$: IF
      X$ = "N" THEN 4000
4080 IF X$ < > "Y" THEN 4070
4090 RETURN

```

Review test question/response data.

```

5000 FOR I = 1 TO QN: HOME :Q$ =
      Q$(I): GOSUB 9000
5010 INVERSE : PRINT I; ". "; T$(T
      );" VALUE="; V: NORMAL
5020 PRINT Q$: PRINT
5030 INVERSE : IF T = 1 THEN PRINT
      "ANSWER: "; NORMAL : PRINT "
      "TF$(TF): GOTO 5120
5040 PRINT "ANSWERS: "; NORMAL : IF
      T = 3 THEN 5080
5050 FOR J = 1 TO N: JJ = A1 + J -
      1: IF J = C THEN INVERSE
5060 PRINT J; ": NORMAL : PRINT ".
      "; A$(JJ): IF PEEK (37) > 1
      8 THEN FLASH : PRINT " "; GOSUB
      10000: NORMAL : PRINT B$; " "
      ; B$;
5070 NEXT J: GOTO 5120
5080 FOR J = 1 TO N: JJ = A1 + J -
      1
5090 L = LEN (A$(JJ)): NC = C: IF
      NC = 0 OR NC > L THEN NC = L
5100 PRINT A$(JJ); ": HTAB 1: INVERSE
      : PRINT LEFT$ (A$(JJ), NC)::
      NORMAL : IF PEEK (37) > 18
      THEN FLASH : PRINT " "; NORMAL
      : GOSUB 10000: PRINT B$; " ";
      B$;
5110 NEXT J
5120 IF HS THEN INVERSE : PRINT
      : PRINT "HINT (PENALTY="HP"/
      10): ": NORMAL : PRINT H$(I)
5130 PRINT : PRINT BR$; " PRESS A
      KEY TO CONTINUE "; BR$: GOSUB
      10000
5140 NEXT I
5150 RETURN

```

Save test file data.

```

6000 HOME : VTAB 5: PRINT "USE F
      ILE NAME 'F$' ?": GOSUB 10
      000: IF X$ = "Y" THEN 6020
6010 GOSUB 2000

```

Check password.

```

6020 IF PW$ > "" THEN PRINT : PRINT
      "USE PASSWORD "; PW$; "?": GOSUB
      10000: IF X$ = "Y" THEN 6040
6030 PRINT : INPUT "PASSWORD FOR
      THIS TEST: "; PW$: IF PW$ =
      "" THEN 6030

```

Encode password.

```

6040 X$ = "00" + STR$ (QN): X$ =
      RIGHT$ (X$, 3): X$ = "00" + STR$
      (AP) + X$: X$ = RIGHT$ (X$, 6
      ): Q$(0) = X$ + PW$

```

Print data to disk.

```

6050 PRINT : PRINT "SAVING 'F$'
      ' TO DISK"
6060 ONERR GOTO 6140
6070 PRINT D$"OPEN"QF$
6080 PRINT D$"DELETE"QF$
6090 PRINT D$"OPEN"QF$
6100 PRINT D$"WRITE"QF$
6110 FOR I = 0 TO QN: PRINT QUD$
      ; Q$(I); QUD$: NEXT I
6120 FOR I = 1 TO AP: PRINT QUD$
      ; A$(I); QUD$: NEXT I
6130 FOR I = 1 TO QN: PRINT QUD$
      ; H$(I); QUD$: NEXT I: GOTO 61
      50
6140 FLASH : PRINT : PRINT "ERRO
      R "; PEEK (222): NORMAL : CALL
      FIX: FOR Z = 1 TO 2500: NEXT
      Z
6150 PRINT D$"CLOSE": POKE 216, 0
      : RETURN

```

Load in old test file data.

```

7000 PRINT : HOME : ONERR GOTO
      7090
7010 PRINT D$"CATALOG": E = 0
7020 PRINT : INPUT "FILE NAME:
      Q. "; F$: QF$ = "Q." + F$
7030 PRINT D$"OPEN"QF$
7040 PRINT D$"READ"QF$

```

Decode password.

```

7050 INPUT Q$(0): QN = VAL ( MID$
      (Q$(0), 4, 3)): AP = VAL ( LEFT$
      (Q$(0), 3)): PW$ = RIGHT$ (Q$
      (0), LEN (Q$(0)) - 6)

```

Read in test file data.

```

7060 FOR I = 1 TO QN: INPUT Q$(I
      ): NEXT I
7070 FOR I = 1 TO AP: INPUT A$(I
      ): NEXT I
7080 FOR I = 1 TO QN: INPUT H$(I
      ): NEXT I: GOTO 7110
7090 E = PEEK (222): IF E = 5 THEN
      PRINT : PRINT BEL$ + BEL$; "
      FILE 'F$" NOT FOUND": PRINT
      D$"DELETE"QF$: GOTO 7100
7095 FLASH : PRINT : PRINT "ERRO
      R "; E: NORMAL
7100 CALL FIX: FOR Z = 1 TO 2500
      : NEXT Z

```

```

7110 PRINT D$"CLOSE": POKE 216, 0
      : RETURN

```

Print out test.

```

8000 VTAB 22: HTAB 1: PRINT "PLE
      ASE TURN ON YOUR PRINTER": PR#
      1: PRINT : HOME
8010 FOR I = 1 TO QN: Q$ = Q$(I)
8020 GOSUB 9000: PRINT I; ". "; Q$
      : PRINT
8030 IF T = 1 THEN PRINT U3$"TR
      UE": PRINT U3$"FALSE": GOTO
      8060
8040 IF T = 3 THEN PRINT U3$U3$
      U3$U3$U3$: GOTO 8060
8050 FOR J = 1 TO N: PRINT U3$; J
      ; ". "; A$(A1 + J - 1): NEXT J
8060 PRINT : PRINT : PRINT : NEXT
      I: PR# 0: RETURN

```

Separate encoded question/response data.

```

9000 T = VAL ( LEFT$ (Q$, 1)): V =
      VAL ( MID$ (Q$, 2, 1))
9010 IF T = 1 THEN TF = VAL ( MID$
      (Q$, 3, 1)): GOTO 9030
9020 A1 = VAL ( MID$ (Q$, 3, 3)): N
      = VAL ( MID$ (Q$, 6, 1)): C =
      VAL ( MID$ (Q$, 7, 1))
9030 X = 9 - 4 * (T = 1): HS = VAL
      ( MID$ (Q$, X - 1, 1)): HP = VAL
      ( MID$ (Q$, X, 1))
9040 Q$ = RIGHT$ (Q$, LEN (Q$) -
      X)
9050 RETURN

```

Exit from program.

```

9900 VTAB 22: HTAB 1: PRINT "
      DO YOU WANT TO QUIT?": GOSUB
      10000: IF X$ = "Y" THEN END
9910 RETURN

```

Single character input routine.

```

10000 POKE CLR, 0: WAIT KEY, PRESS
      : GET X$: RETURN

```

Read menu heading and options.

```

11000 READ N$: READ N: FOR I = 1
      TO N: READ N$(I): NEXT I

```

Print headings.

```

11010 INVERSE : HTAB 4: PRINT N$
      : NORMAL : V = PEEK (37)

```

Print options.

```

11020 FOR I = 1 TO N
11030 PRINT TAB( 4)I; ". "; N$(I)
11040 POKE S, PEEK (S)
11050 NEXT I

```

Display menu prompt.

```

11060 VTAB V: HTAB 1: PRINT "="
      ;: POKE CLR,0
11070 GOSUB 10000:IT = VAL (X%)
11080 POKE S, PEEK (S)
11090 IF IT < 1 OR IT > N THEN 1
      1060
11100 VTAB V: HTAB 1: PRINT " "
      ;:VV = V + IT: VTAB VV: HTAB
      1: PRINT "=>";
11110 GOSUB 10000:X = ASC (X%)
11120 IF X < > 13 THEN IT = VAL
      (X%): VTAB VV: HTAB 1: PRINT
      " ";: GOTO 11080
11130 VTAB V + N + 2: HTAB 1: RETURN

```

Poke in stack-fix routine.

```

15000 X% = "104,168,104,166,223,1
      54,072,152,072,096"
15010 FIX = 768: FOR I = 0 TO 9:J
      = VAL ( MID$(X%,I * 4 + 1
      ,3)): POKE FIX + I,J: NEXT
15020 RETURN

```

Data for question options.

```

20000 DATA " QUESTION TYPES ",4,
      MIXED TYPES,TRUE/FALSE,MULTI
      PLE CHOICE,FILL-IN
20010 DATA " QUESTION VALUES ",2
      ,ALL THE SAME,VARIABLE
20020 DATA " HINT OPTION ",2,ON,
      OFF
20030 DATA " HINT PENALTY ",2,FI
      XED PERCENTAGE,VARIABLE PERC
      ENTAGE

```

**APPLE™ SWAT TABLE FOR:
CATS DISK VERSION**

LINES	SWAT CODE	LENGTH
10 - 200	JD	269
210 - 590	SD	488
600 - 1110	IF	400
1120 - 2040	FH	523
2050 - 3050	MJ	435
3060 - 3170	YJ	273
3180 - 3290	OU	388
3300 - 3540	ZH	330
3550 - 4030	NO	212
4040 - 5050	PY	365
5060 - 6010	HY	390
6020 - 6130	AN	397
6140 - 7090	PH	397
7095 - 9010	WD	349
9020 - 11040	UX	311
11050 - 15020	PD	303
20000 - 20030	DN	213

```

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$   APPLESOFT BASIC   $
$ 'CATS - TAPE CHANGES' $
$   AUTHOR: JON R. VOSKUIJL $
$   (C) 1982   SOFTSIDE $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

```

These are the changes necessary to adapt CATS for use with cassette.

DELETE lines 2000 to 2080

Change the following three lines as shown.

```

1030 N*(3) = "SAVE TEST TO TAPE"
1050 N*(5) = "LOAD A TEST FROM TA
      PE"
3005 REM

```

These lines should replace lines 6000 to 6150 of the disk version.

```

6000 HOME : VTAB 6
6010 IF PW$ > "" THEN PRINT "US
      E PASSWORD ";PW$;" ?": GOSUB
      10000: IF X% = "Y" THEN 6030
6020 PRINT : INPUT "PASSWORD FOR
      THIS TEST: ";PW$: IF PW$ =
      "" THEN 6020
6030 X% = "00" + STR$(QN):X% =
      RIGHT$(X%,3):X% = "00" + STR$(
      AP) + X%:X% = RIGHT$(X%,6
      ):Q$(0) = X% + PW$
6040 PRINT : INPUT "POSITION TAP
      E, START RECORDING, THEN
      PRESS 'RETURN'.":X%
6050 PRINT : PRINT "SAVING"
6060 STORE Q$: STORE A$: STORE H
      $: CALL WR: RETURN

```

These lines should replace lines 7000 to 7110 of the disk version.

```

7000 PRINT : HOME : VTAB 6
7010 INPUT "POSITION TAPE, START
      PLAYING, THEN PRESS 'R
      ETURN'.":X%
7020 FLASH : PRINT : PRINT "LOAD
      ING": NORMAL
7030 RECALL Q$: RECALL A$: RECALL
      H$: CALL RD
7040 CALL FIX,Q$(0): CALL FIX,A$
      (0): CALL FIX,H$(0)
7050 QN = VAL ( MID$( Q$(0),4,3)
      ):AP = VAL ( LEFT$( Q$(0),3
      )):PW$ = RIGHT$( Q$(0), LEN
      (Q$(0)) - 6): RETURN

```

These lines should replace lines 15000 to 15020 of the disk version.

```

15000 WR = 768:RD = 800:FIX = 865
      :M = WR
15010 X% = "160,111,162,000,132,0
      60,134,061,160,116,032,025,0
      03,164,111,166,112,132,060,1
      34,061,164,115,166,116,132,0
      62,134,063,076,205,254,160,0
      24,162,000,132,060,134,061,1
      60,029,132,062,134,063,032,2
      53,254,165,028,056,229,024,1
      33"

```

```

15020 GOSUB 15100
15030 X% = "008,165,029,229,025,1
      33,009,160,002,162,000,056,1
      81,111,149,062,245,008,149,1
      11,149,060,232,136,208,242,1
      65,024,056,229,111,133,006,1
      65,025,229,112,133,007,076,2
      53,254,032,190,222,032,123,2
      21,032,108,221,160,002,165,1
      55"

```

```

15040 GOSUB 15100
15050 X% = "024,113,155,133,008,2
      00,165,156,113,155,133,009,1
      60,000,177,160,240,015,200,1
      77,160,056,229,006,145,160,2
      00,177,160,229,007,145,160,1
      69,003,024,101,160,133,160,1
      44,002,230,161,197,008,208,2
      20,165,009,197,161,208,214,0
      96"

```

```

15060 GOSUB 15100: RETURN
15100 FOR I = 0 TO 54:J = VAL (
      MID$(X%,I * 4 + 1,3)): POKE
      M + I,J: NEXT :M = M + 55: RETURN

```

**APPLE™ SWAT TABLE FOR:
CATS TAPE VERSION**

LINES	SWAT CODE	LENGTH
10 - 200	JD	269
210 - 590	SD	488
600 - 1110	ID	400
1120 - 3040	OI	553
3045 - 3150	TX	261
3160 - 3270	YC	389
3280 - 3520	EV	325
3530 - 4010	AT	226
4020 - 5030	VE	352
5040 - 5150	RU	381
6000 - 6140	AR	460
6150 - 8040	NC	404
8050 - 11000	YY	362
11010 - 11120	FF	250
11130 - 15030	KA	516
15040 - 20030	JZ	515

TRS-80®

```

#####
$ TRS-80 Model I/III BASIC $
$ "CATS" $
$ Author: Jon R. Voskuil $
$ Translated: Alan J. Zett $
$ (c) 1982, SoftSide $
#####

```

10 GOTO500

Custom line input routine.

```

100 I$="":L=PEEK(16416)+PEEK(16417)*256-15360:V=INT(L/64):H=L-64
    #V
110 PRINTCHR$(14);:C$=INKEY$:IFC$=" "THEN110ELSEC=ASC(C$)
115 IFC<32THEN150
120 IFC=34THENC=39:C$=""
130 IFL<230THEN110
140 I$=I$+C$:PRINTC$;:GOTO110

```

Process control commands.

```

150 IFC=13THENPRINTCHR$(15);:RETURN
160 IFC=21THENC=95:C$=CHR$(C):GOTO130
170 IFC=5THENPRINT@V#64+H,CHR$(31);:PRINT@V#64+H,;:GOTO100
180 IFC<>8THEN110
190 L=LEN(I$):IFL<2THENI$="":GOTO210
200 I$=LEFT$(I$,L-1)
210 IFL>0THENPRINTC$ "C$;
220 GOTO110

```

Display title page and initialize program.

```

500 CLS:PRINTCHR$(23)
510 PRINT@152,"C A T S"
520 PRINT@270,"BY JON R. VOSKUIL"
525 PRINT@388,"TRANSLATION BY ALAN J. ZETT"
530 PRINT@650,"QUESTION ENTRY MODULE"

```

Calculate free string space.

```

540 CLEAR 0:DIM Q$(100),A$(300),H$(100),TF$(1),T$(3),N(9):IFMEM#
    .9>32767THENCLEAR32767ELSECLEARMEM#.9

```

Define variables.

```

550 DEFINITA-D,F,G,I-U,W-Z:A1=0:AP=0:CC=0:CM=0:EO=0:ED=0:H=0:HP=0
:HS=0:HY=0:I=0:IT=0:J=0:JJ=0:L=0:N=0:NC=0:NN=0:PF=0:QN=0:QV=0:T=
0:TF=0:TY=0:V=0:VF=0:VV=0:X=0:Z=0:B$=CHR$(8):C$="":F$="":I$="":P
W$="":Q$="":QF$="":QUO$=CHR$(34):U3$=STRING$(3,95):X$=""

```

Adjust cursor for Model I or III.

```

555 IFPEEK(293)=73THENPRINTCHR$(21);:U$=CHR$(244)+CHR$(245)+CHR$
(246)ELSEU$="=>"

```

Assign string arrays.

```

560 DIM Q$(100),A$(300),H$(100)
570 T$(1)="TRUE OR FALSE":T$(2)="MULTIPLE CHOICE":T$(3)="FILL-IN"
    ":TF$(0)="FALSE":TF$(1)="TRUE"
590 QN=0:AP=0
600 PRINT@966,"PRESS ANY KEY TO CONTINUE";:GOSUB10000

```

Main program loop.

```

1000 CLS:V=3:H=0:N$="QUESTION ENTRY MAIN OPTIONS MENU":N=8
1010 N$(1)="ADD TO TEST IN MEMORY"
1020 N$(2)="REVIEW TEST IN MEMORY"
1030 N$(3)="SAVE TEST TO DISK"

```

QUESTION TYPES

- 1 - MIXED TYPES
- 2 - TRUE/FALSE
- 3 - MULTIPLE CHOICE
- 4 - FILL-IN

QUESTION VALUES

- 1 - ALL THE SAME
- 2 - VARIABLE

HINT OPTION

- 1 - ON
- 2 - OFF

HINT PENALTY

- 1 - FIXED PERCENTAGE
- 2 - VARIABLE PERCENTAGE

HINT PENALTY IN TENTHS? (1-9) 5

ALL OK?

```

1040 N$(4)="BEGIN A NEW TEST"
1050 N$(5)="LOAD A TEST FROM DISK"
1060 N$(6)="RESET QUESTION ENTRY OPTIONS"
1070 N$(7)="PRINT TEST ON PRINTER"
1080 N$(8)="QUIT"
1090 ONERRORGOTO0
1100 V=3:GOSUB11010
1110 IFIT<40RIT>50RQN=0THEN1140
1120 PRINT " ERASES TEST IN MEMORY. OK? ";:GOSUB10000:IFX$("<
>Y"THEN1000

```

Erase old test.

```

1130 QN=0:AP=0:PRINT"ERASING TEST ...";:FORX=1TD100:Q$(X)=""<
>A$(X)=""<
>A$(X+100)=""<
>A$(X+200)=""<
>H$(X)=""<
>NEXT
1140 IF(IT<40RIT=7)ANDQN=0THENPRINT "==== NO TEST IN MEMORY =
====":FORZ=1TD700:NEXT:GOTO1000
1150 DNITGOSUB3000,5000,6000,3000,7000,4000,8000,9900
1160 GOTO1000

```

Get file name and test for existing file.

```

2000 CLS:PRINT:INPUT"FILE NAME FOR THIS TEST";F$:QF$=F$+"/Q"
2010 PRINT:PRINT"INSERT STORAGE DISK AND PRESS A KEY";:GOSUB10000
2020 ONERRORGOTO2060
2030 OPEN"1",1,QF$
2040 CLOSE:PRINT:PRINT"QUIZ FILE "QUO$F$QUO$" ALREADY EXISTS ON
THIS DISK.";PRINT:PRINT"OK TO ERASE? (Y/N)":GOSUB10000:IFX$="Y"
HEN2080ELSE2000

```

Trap for "File Not Found" error.

```

2060 E=ERR/2-1:IFE=52THENRESUME2080
2070 CLOSE:PRINT:PRINT"==== ERROR"E IN LINE"ERL"====":FORZ=1TD13
00:NEXT:RESUME2080
2080 CLOSE:RETURN

```

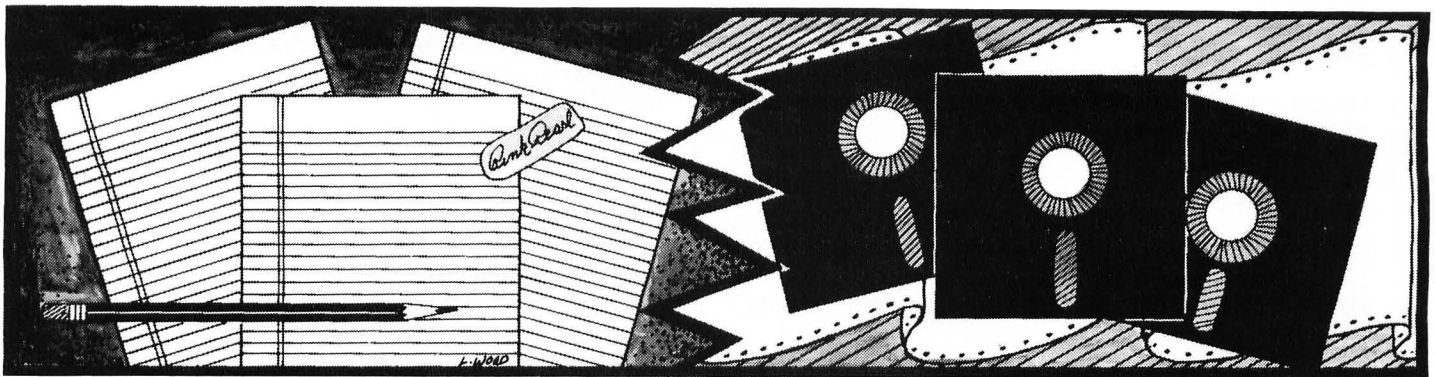
Check free question space. Cancel input if no more room.

```

3000 CLS:PRINT"CHECKING SPACE ...":IFFRE(A$)<766THENPRINT:PRINT"
# NO MORE ROOM FOR QUESTIONS #":FORZ=1TD1300:NEXT:GOTO1000

```

If not set, get test file name and question options.



```
3005 IFQN=0THENGOSUB2000
3010 IFED=0THENGOSUB4000
3020 QV=1
```

Input questions and type chosen.

```
3030 A1=AP+1:CLS:PRINT"QUESTION NUMBER"QN+1
3040 PRINTQ128,,:T=TYPE:IFT=4THENPRINT"TYPE? 1=T/F, 2=MULTI, 3
=FILL-IN, 0=MENU";:GOSUB10000:T=VAL(X%):IFX%="0"ORX%=CHR$(13)THE
NRETURN
3050 IFT<1ORT>3THEN3040
3060 PRINTQ32,T$(T):PRINTQ128,CHR$(31);
3070 GOSUB100
3080 IFI$=""THEN3620
3090 QN=QN+1:Q$(QN)=I$
```

Input questions responses.

```
3100 PRINT:PRINT:IFT<>2THEN3160
3110 PRINT"ANSWER CHOICES:"NC=1
3120 PRINTNC"- ";:GOSUB100
3130 IFI$=""THENPRINTQV*64,CHR$(31):NC=NC-1:GOTO3160
3140 AP=AP+1:A$(AP)=I$
```

```
3150 PRINT:IFNC<9THENNC=NC+1:GOTO3120
3160 PRINT"CORRECT RESPONSE:"
3170 ONT60T03180,3210,3230
```

Input True/False response.

```
3180 PRINT"TRUE OR FALSE? (T/F) ";:GOSUB10000:PRINTB%X%:X%=LEFT
$(X$,1):IFX%<"T"ANDX%<"F"THENPRINTCHR$(27);:GOTO3180
3190 TF=0:IFX%="T"THENTF=1
3200 GOTO3280
3210 PRINT"NUMBER? ";:GOSUB10000:PRINTB%X%:CC=VAL(X%):IFCC<10RC
C>NCTHENPRINTCHR$(27);:GOTO3210
3220 GOTO3280
```

```
3230 NN=0:GOSUB100:PRINT
3240 NN=NN+1:AP=AP+1:A$(AP)=I$:IFNN=9THEN3270
```

Select an alternate response.

```
3250 PRINT"ALTERNATE ANSWER (RETURN IF NONE):":GOSUB100:PRINT
3260 IFI$<">"THEN3240ELSEPRINTCHR$(27);
3270 PRINT"CHARS TO MATCH? (1-9, RETURN=ALL) ";:GOSUB10000:CM=V
AL(X%):IFCM<10RCM>9THENCM=0
3275 IFCM=0THENPRINTB$*ALL"ELSEPRINTB$*B$CM
3280 IFHYNT=0THEN3350
```

Input hint if applicable.

```
3290 PRINT"HINT (RETURN IF NONE):"
3300 GOSUB100:PRINT
3310 H$(QN)=I$
3320 HS=1:IFI$=""THENHS=0:PRINTCHR$(27);:GOTO3350
3330 IFPFLAG=0THEN3350
```

Input hint penalty and question value.

```
3340 PRINT"HINT PENALTY IN TENTHS (1-9) ";:GOSUB10000:PRINTB%X%
:HP=INT(VAL(X%)+.5):IFHP<10RHP>9THENPRINTCHR$(27);:GOTO3340
```

```
3350 IFVFLAG=0THEN3500
```

```
3360 PRINT"QUESTION VALUE? (1-9) ";:GOSUB10000:PRINTB%X%:QV=VAL
(X%):IFQV<10RQV>9THENPRINTCHR$(27);:GOTO3360
```

Construct encoded question/response data.

```
3500 Q%=RIGHT$(STR$(T),1)
3510 Q%=Q%+RIGHT$(STR$(QV),1)
3520 IFT=1THENQ%=Q%+RIGHT$(STR$(TF),1):GOTO3580
3530 X%="00"+RIGHT$(STR$(A1),LEN(STR$(A1))-1):X%=RIGHT$(X%,3):Q%
=Q%+X%
3540 X=NC:IFT=3THENX=NN
3550 Q%=Q%+RIGHT$(STR$(X),1)
3560 X=CC:IFT=3THENX=CM
3570 Q%=Q%+RIGHT$(STR$(X),1)
3580 Q%=Q%+RIGHT$(STR$(HS),1)
3590 Q%=Q%+RIGHT$(STR$(HP),1)
3600 Q$(QN)=Q%+Q$(QN)
3610 GOTO3000
3620 RETURN
```

Select question options.

```
4000 CLS:RESTORE:EQ=1:V=0
4010 GOSUB11000:TYPE=IT-1:IFTYPE=0THENTYPE=4
4020 V=0.5:GOSUB11000:VFLAG=IT-1
4030 V=7:GOSUB11000:HYNT=2-IT
4040 IFHYNT=0THEN4070
4050 V=7.5:GOSUB11000:PFLAG=1:IFIT=2THEN4070
4060 PFLAG=0:PRINTQ768,"HINT PENALTY IN TENTHS? (1-9) ";:GOSUB1
0000:PRINTB%X%:HP=INT(VAL(X%)+.5):IFHP<10RHP>9THEN4060
4070 PRINTQ960,"ALL OK?";:GOSUB10000
4080 IFX%="N"THEN4000ELSEIFX%<"Y"THEN4070ELSEReturn
```

Review test question/response data.

```
5000 FORI=1TOQN:CLS:Q%=Q$(I):GOSUB9000
5010 PRINTI"- T$(T) / VALUE ="V
5020 PRINT:PRINTQ%:PRINT
5030 IFT=1THENPRINT"ANSWER: "TF$(TF):GOTO5120
5040 PRINT"ANSWERS: ";:IFT=3THEN50B0ELSEPRINT
5050 FORJ=1TON:JJ=A1+J-1
5060 PRINTRIGHT$(CHR$(32-10*(J=C))+RIGHT$(STR$(J),1),2)". "A$(JJ
):IFPEEK(16417)>62THENPRINT"?";:GOSUB10000:PRINTB$* "B$;
5070 NEXT:GOTO5120
```

Calculate portion of response to match.

```
5080 PRINT"(MUST MATCH";:IFC=0THENPRINT" ALL ";ELSEPRINTC;
5090 PRINT"CHARACTERS");:FORJ=1TON:JJ=A1+J-1
5100 PRINTA$(JJ):IFPEEK(16417)>62THENPRINT"?";:GOSUB10000:PRINTB
$* "B$;
5110 NEXT
5120 IFHS<>0THENPRINT:PRINT"HINT: (PENALTY ="HP"/ 10)":PRINTH$(I
)
5130 PRINT:PRINT"PRESS A KEY TO CONTINUE";:GOSUB10000
5140 NEXT
```

5150 RETURN

Save test file data.

```
6000 CLS:PRINT@192,"USE FILE NAME "QUO$F$QUO$ "?:GOSUB10000:IFX$="Y"THEN6020
6010 GOSUB2000
```

Check password.

```
6020 IFPW$>"THENPRINT:PRINT"USE PASSWORD "QUO$PW$QUO$ "?:GOSUB
10000:IFX$="Y"THEN6040
6030 PRINT:INPUT"PASSWORD FOR THIS TEST";PW$:IFPW$=""THEN6030
```

Encode password.

```
6040 X$=RIGHT$("00"+RIGHT$(STR$(QN),LEN(STR$(QN))-1),3):Q$(0)=RI
GHT$("00"+RIGHT$(STR$(AP),LEN(STR$(AP))-1)+X$,6)+PW$
6050 PRINT:PRINT"SAVING "QUO$F$QUO$ " TO DISK"
6060 ONERRORGOTO6140
```

Print data to disk.

```
6070 OPEN"D",1,QF$
6110 FORI=0TOQN:PRINT#1,QUO$Q$(I)QUO$:NEXT
6120 FORI=1TOAP:PRINT#1,QUO$A$(I)QUO$:NEXT
6130 FORI=1TOQN:PRINT#1,QUO$H$(I)QUO$:NEXT:GOTO6150
6140 CLOSE:PRINT:PRINT"==== ERROR"ERR/2-1"IN LINE"ERL"====":FORZ
=1TO1300:NEXT:RESUME6150
6150 CLOSE:RETURN
```

Load in old test file data.

```
7000 CLS
7010 ONERRORGOTO7090
7020 PRINT@192,"INPUT FILE NAME";:INPUTF$:QF$=F$+"/"+Q$
7030 OPEN"I",1,QF$
```

Decode password.

```
7050 INPUT#1,Q$(0):QN=VAL(MID$(Q$(0),4,3)):AP=VAL(LEFT$(Q$(0),3)
):PW$=RIGHT$(Q$(0),LEN(Q$(0))-6)
7060 FORI=1TOQN:INPUT#1,Q$(I):NEXT
7070 FORI=1TOAP:INPUT#1,A$(I):NEXT
7080 FORI=1TOQN:INPUT#1,H$(I):NEXT:GOTO7100
7090 E=ERR/2-1:IFE=52THENPRINT:PRINT"==== FILE NOT FOUND ====":F
ORZ=1TO1300:NEXT:RESUME7100
7095 CLOSE:PRINT:PRINT"==== ERROR"E"IN LINE"ERL"====":FORZ=1TO13
00:NEXT:RESUME7100
7100 CLOSE:RETURN
```

Print out test.

```
8000 PRINT@960,"PLEASE TURN ON YOUR PRINTER";:LPRINT " ":PRINT@96
0,"STAND BY, PRINTING TEST ...";
8010 FORI=1TOQN:Q$=Q$(I)
8020 GOSUB9000:LPRINTI"- "Q$:LPRINT "
8030 IFT=1THENLPRINTU3$"TRUE":LPRINTU3$"FALSE":GOTO8060
8040 IFT=3THENLPRINTU3$U3$U3$U3$:GOTO8060
8050 FORJ=1TON:LPRINTU3$J"- "A$(A1+J-1):NEXT
8060 LPRINT " ":LPRINT " ":LPRINT " ":NEXT:RETURN
```

Separate encoded question/response data.

```
9000 T=VAL(LEFT$(Q$,1)):V=VAL(MID$(Q$,2,1))
9010 IFT=1THENTF=VAL(MID$(Q$,3,1)):GOTO9030
9020 A1=VAL(MID$(Q$,3,3)):N=VAL(MID$(Q$,6,1)):C=VAL(MID$(Q$,7,1)
)
9030 X=9+4*(T=1):HS=VAL(MID$(Q$,X-1,1)):HP=VAL(MID$(Q$,X,1))
9040 Q$=RIGHT$(Q$,LEN(Q$)-X)
9050 RETURN
```

Exit from program.

```
9900 PRINT@996,"DO YOU WANT TO QUIT?":GOSUB10000:IFX$="Y"THENCLS
:PRINTCHR$(21);:END
9910 RETURN
```

Single character input routine.

```
10000 X$=INKEY$:IFX$=""THEN10000ELSERETURN
```

Read menu heading and options.

```
11000 READN$:READN:FORI=1TON:READN$(I):NEXT
```

Print headings.

```
11010 PRINT@V$64+4,CHR$(143)" N$ "CHR$(143);
```

Print options.

```
11020 FORI=1TON
11030 PRINT@V$(V+I)$64+3,I"- "N$(I);
11050 NEXT
```

Display menu prompt.

```
11060 PRINT@V$64,U$;
11070 GOSUB10000:IT=VAL(X$)
11090 IFIT<1ORIT>NTHEN11060
11100 PRINT@V$64," ";:VV=V+IT:PRINT@VV$64,U$;
11110 GOSUB10000:X=ASC(X$)
11120 IFX<>13THENIT=VAL(X$):PRINT@VV$64," ";:GOTO11090
11130 PRINT@V$(V+N+2)$64,:RETURN
```

Data for question options.

```
20000 DATA QUESTION TYPES,4,MIXED TYPES,TRUE/FALSE,MULTIPLE CHOI
CE,FILL-IN
20010 DATA QUESTION VALUES,2,ALL THE SAME,VARIABLE
20020 DATA HINT OPTION,2,ON,OFF
20030 DATA HINT PENALTY,2,FIXED PERCENTAGE,VARIABLE PERCENTAGE
```

TRS-80® SWAT TABLE FOR: CATS DISK VERSION

LINES	SWAT CODE	LENGTH
10 - 190	DV	314
200 - 555	PX	548
560 - 1070	VA	477
1080 - 2020	RC	475
2030 - 3050	BX	510
3060 - 3170	EU	273
3180 - 3280	GE	447
3290 - 3530	NJ	419
3540 - 4020	HX	238
4030 - 5050	UB	395
5060 - 6010	YB	398
6020 - 7000	WB	436
7010 - 8010	ZE	459
8020 - 9900	ZN	400
9910 - 11110	AT	252
11120 - 20030	FR	264


```

#####
$ TRS-80 Model I/III BASIC $
$ "CATS: TAPE CHANGES" $
$ Author: Alan J. Zett $
$ (c) 1982, SoftSide $
#####

```

```

1030 N$(3)="SAVE TEST TO TAPE"
1050 N$(5)="LOAD A TEST FROM TAPE"
3005 REM
6000 CLS
6010 PRINT@192,:
6020 IFPW$=""THENPRINT"USE PASSWORD "QUO$PW$QUO$ "?:GOSUB10000:
IFX$="Y"THEN6040
6030 PRINT:INPUT"PASSWORD FOR THIS TEST";PW$:IFPW$=""THEN6030
6040 X$=RIGHT$("00"+RIGHT$(STR$(QN),LEN(STR$(QN))-1),3):Q$(0)=RI
GHT$("00"+RIGHT$(STR$(AP),LEN(STR$(AP))-1)+X$,6)+PW$
6050 PRINT:PRINT"POSITION TAPE, THEN PRESS <ENTER> TO SAVE":GOSU
B10000
6060 PRINT:PRINT"SAVING *";
6070 L=PEEK(16416)+PEEK(16417)*256-15360
6110 FORI=0TOQN:GOSUB7095:PRINT#-1,QUO$Q$(I)QUO$:NEXT
6120 FORI=1TOAP:GOSUB7095:PRINT#-1,QUO$A$(I)QUO$:NEXT
6130 FORI=1TOQN:GOSUB7095:PRINT#-1,QUO$H$(I)QUO$:NEXT
6140 RETURN
6150 REM
7000 CLS
7010 PRINT@192,"POSITION TAPE, THEN PRESS <ENTER> TO LOAD":GOSUB
10000
7020 PRINT:PRINT"LOADING *";
7030 L=PEEK(16416)+PEEK(16417)*256-15360
7050 INPUT#-1,Q$(0):QN=VAL(MID$(Q$(0),4,3)):AP=VAL(LEFT$(Q$(0),3
)):PW$=RIGHT$(Q$(0),LEN(Q$(0))-6)
7060 FORI=1TOQN:GOSUB7095:INPUT#-1,Q$(I):NEXT
7070 FORI=1TOAP:GOSUB7095:INPUT#-1,A$(I):NEXT
7080 FORI=1TOQN:GOSUB7095:INPUT#-1,H$(I):NEXT
7090 RETURN
7095 PRINT@L,CHR$(42+10*(INT(I/2)*2=I));
7100 RETURN

```

TRS-80® SWAT TABLE FOR:
CATS TAPE VERSION

LINES	SWAT CODE	LENGTH
10 - 190	DV	314
200 - 555	PX	548
560 - 1070	UY	477
1080 - 3010	LT	474
3020 - 3130	QA	346
3140 - 3250	QW	395
3260 - 3360	HH	444
3500 - 3610	RD	273
3620 - 5010	FX	384
5020 - 5130	HP	411
5140 - 6120	HL	412
6130 - 7090	XV	346
7095 - 9020	VC	404
9030 - 11060	DN	291
11070 - 20030	GE	362

ATTENTION AUTHORS

SoftSide Publications is actively seeking program, article and review submissions for the TRS-80®, Apple™ and ATARI® home computers.

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

Please be sure to include full documentation of subroutines and a list of variables, also 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.

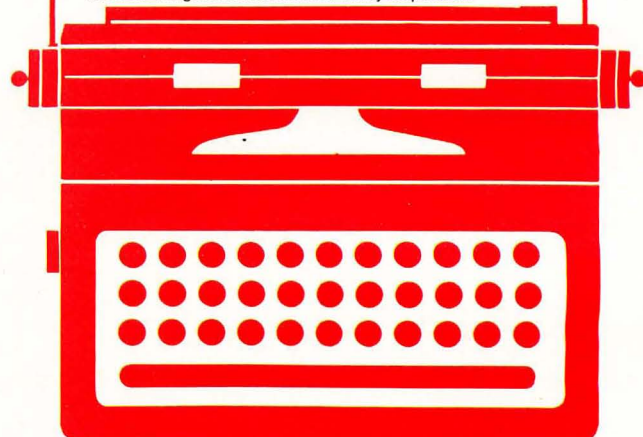
Please be sure to pack your cassettes and disks carefully and to include your return address and phone number.

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.



“Simply the best word processor...anywhere”

For the Apple™

No hardware additions needed — What you see is what you get. All functions are displayed on the screen exactly as they appear in print including:

- Underlining
- Bold
- Superscript
- Even/normal justification
- Lower and upper case
- Block movement
- Global replace
- Plus many more features



“Word Handler is simply the best word processing software I could find anywhere by far.”
William R. Moroney
President
Electronic Funds
Transfer Association

Now Available!
List Handler
A mailing list program to work with the Word Handler (interfaces with VisiCalc™ and DB Master™) stores up to 3000 records per disk, unlimited sorting fields.

Word Handler

Once you buy a Word Handler we don't forget you; our customer service department is available daily. We support our products.

When upgrades are introduced you are supplied with a replacement disk FREE! Contact your local Apple™ dealer for a demonstration. You'll be glad you did.

Silicon Valley Systems, Inc. 1625 El Camino Real #4 Belmont, CA 94002 (415) 593-4344

Silicon Valley Systems

GENERAL INFORMATION

Concerning SoftSide line listings, SWAT & Magnetic Media

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

SWAT TABLES

At the conclusion of each line listing of a *SoftSide* program, we include a *SWAT* (*Strategic Weapon Against Typos*) Table. *SWAT* was published in issue #30 of *SoftSide* and is available as a free reprint. Please send a self-addressed, stamped envelope to *SoftSide* Publications, Inc., Dept. *SWAT*, 6 South Street, Milford, NH 03055.

APPLE™

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

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

ATARI®

Line Listings use the following conventions in representing unprintable characters, unless otherwise noted:

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

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

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

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

SWAT — Before appending *SWAT* to a program in memory, the program to be *SWATed* must first be *LISTed* to disk or cassette (using *LIST "D:FILENAME"* for disk or *LIST "C:"* for tape). Next, turn the computer off, then on again, to clear the system and *ENTER* the program back into

memory (using *ENTER "D:filename"* for disk or *ENTER "C:"* for tape). Because of the unique method in which *ATARI® BASIC* stores variables in a program, the variable table must always be in the same order to produce accurate *SWAT* codes. *LISTing* and *ENTERing* the program is the only known way to rebuild the variable table in a specific order so that *SWAT* codes can match.

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

Tapes *CLOAD* in the normal manner. If you have difficulty, try this procedure:

(1) Type *POKE 54018,54* and press *RETURN*.

(2) Turn up the volume on your TV.

(3) Type *CLOAD* and press *RETURN* once.

(4) Press the *PLAY* button and listen.

(5) When you hear a steady lead-in tone, press *RETURN* again.

Side two of the tape is a duplicate of side one.

TRS-80®

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

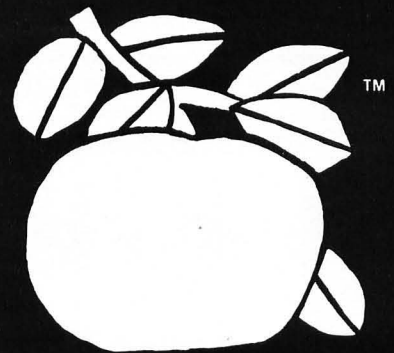
Tapes *CLOAD* in the normal manner on Model I's, and at low speed (500 baud) on Model III's. The first program is a cover/menu program. Side two of the tape is a duplicate of side one.

NOTES ABOUT MAGNETIC MEDIA

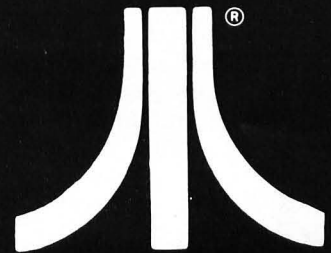
SoftSide disks and tapes are duplicated by reliable, professional duplication services; bad copies are very rare. However, the trip through the mail occasionally wreaks havoc with sensitive magnetic media. If, after a reasonable number of tries and a careful check and cleaning of your equipment, you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

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

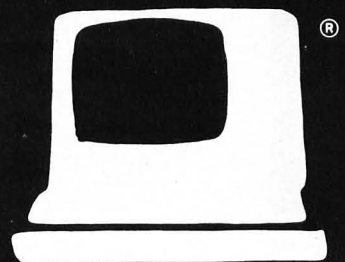
SoftSide



APPLE™/SIDE
page___ 42



ATARI®/SIDE
page___ 58



TRS-80®/SIDE
page___ 79

K-Byter

Snake Scramble

An Apple™ K-Byter by Leonard Vincent, W. Hollywood, CA

In this fast-moving action game, you are a hungry snake trying to gobble up fallen apples. You use the traditional I-J-K-M diamond of keys to turn up, down, left, and right. Every time you take a "byte" out of an apple, you score points and another apple falls somewhere on the playfield. Be careful. If you cross your own path or hit the walls, the game is over. You can cross your path only if an apple has landed on it.

```

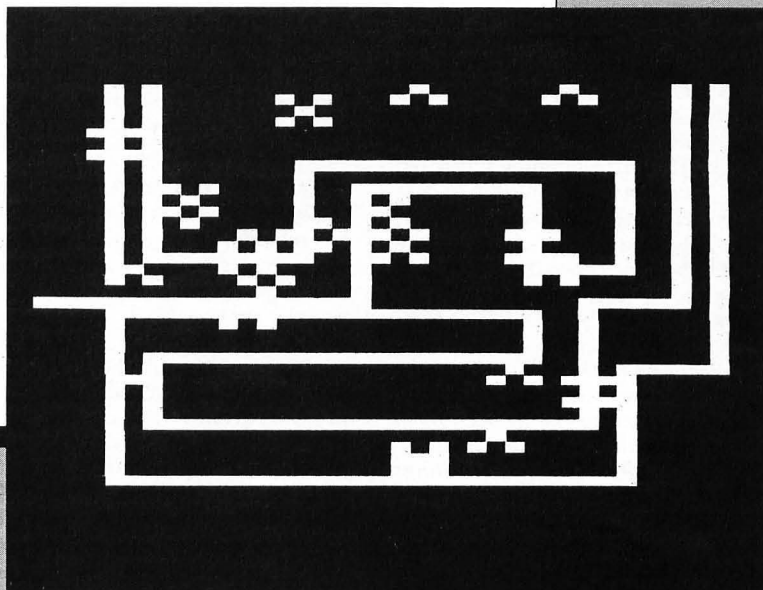
10 GR : POKE - 16302,0: CALL -
    1998:ZX = - 1:ZY = 0:X = 38
    :Y = 20: COLOR= 2: HLIN 0,39
    AT 0: HLIN 0,39 AT 47: VLIN
    0,47 AT 0: VLIN 0,47 AT 39
20 A = RND (1) * 30 + 5:B = RND
    (1) * 40 + 3: COLOR= 15: PLOT
    A,B: COLOR= 13: PLOT A - 1,B
    - 1: PLOT A + 1,B + 1: PLOT
    A - 1,B + 1: PLOT A + 1,B -
    1
30 COLOR= 1:Z = SCRN( X,Y): IF
    Z = 2 OR Z = 1 THEN POKE -
    16368,0: GOTO 100
40 PLOT X,Y: IF Z = 15 OR Z = 13
    THEN 110
50 K = PEEK ( - 16384): IF K < 1
    27 THEN 90
60 K = K - 128:A$ = CHR$ (K): IF
    A$ = "I" THEN ZY = - 1:ZX =
    0

```

```

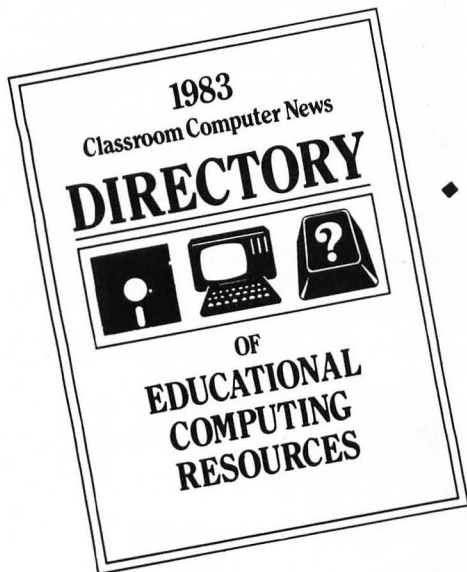
70 IF A$ = "M" THEN ZY = 1:ZX =
    0
80 IF A$ = "K" THEN ZY = 0:ZX =
    1
83 IF A$ = "J" THEN ZY = 0:ZX =
    - 1
85 IF A$ = CHR$ (32) THEN X = RND
    (1) * 35 + 2:Y = RND (1) *
    40 + 5: POKE - 16368,0
90 X = X + ZX:Y = Y + ZY:SC = SC +
    1: FOR T = 1 TO 2:S = PEEK
    ( - 16336): NEXT T: GOTO 30
100 TEXT : HOME : VTAB 12:PA$ =
    "SCORE - " + STR$ (SC): HTAB
    20 - LEN (PA$) / 2: PRINT "
    SCORE - ";: INVERSE : PRINT
    SC: NORMAL : PRINT : HTAB 7:
    PRINT "PRESS ANY KEY TO CON
    TINUE";: GET A$: RUN
110 X = X + ZX:Y = Y + ZY: PRINT
    CHR$ (7);:SC = SC + 1000: FOR
    T = 1 TO 10:S = PEEK ( - 16
    336): NEXT T: GOTO 20

```



The Latest Word...

CCN provides news from the world of educational computing almost as quickly as it happens. Six issues from September to June, bring you the latest in developments in software, hardware and educational applications. Every issue is filled with program listings, software reviews, interviews, news, and commentary providing educators with ideas and practical suggestions for using computers.



...and the Last Word

The 1983 Classroom Computer News Directory of Educational Computing Resources is the most complete, up-to-date listing of people, places, and things needed by computer-using educators. Compiled over the course of a year and updated and checked for accuracy immediately before publication, this 150 plus page volume contains hundreds of thoroughly annotated educational computing resources and thousands of educational computing listings, arranged for quick and easy use.

In A Money Saving Offer.

Order a subscription to CCN now and you may also order a copy of the directory at 30% off or more. See price information below and fill out the coupon at right to subscribe to CCN and reserve your copy of the Directory at this special low rate.

SPECIAL DIRECTORY PRICES WITH CCN SUBSCRIPTION

1 YEAR, 6 ISSUES, U.S.		
no. of subs.	per sub.	per Directory
1 - 10	\$16	\$6.98
11 - 20*	\$13	\$4.98
21 - 40*	\$11	\$4.50
41 or more*	\$8	\$4.25

These prices are in addition to the prices listed above
 For Canada and Foreign (surface mail) add \$5 per subscription and \$2 per Directory
 Payments must be in U.S. dollars drawn on a U.S. bank
 *Plus 1 FREE subscription for your personal use for every order of 11 or more subscriptions.
 List recipients of multiple orders on separate sheet.

Hardcover copies available at \$29.95 each including postage & handling

Yes, I would like to subscribe to *Classroom Computer News* for one year and receive a discount on the 1983 Directory due to be mailed in August. One Directory May Be Ordered Per Subscription.

New Order Renewal Check enclosed Bill me/my school
 ___ Subs @ \$_____ ea. and ___ Directory @ \$_____ ea. \$_____ total
 ___ Hardcover copy @ \$29.95 ea. \$_____ total

List recipients of multiple orders on separate sheet.
 Payments must be in U.S. dollars drawn on a U.S. bank
 Make checks payable to: **Classroom Computer News**
 51 Spring St., Watertown, MA 02172

Name _____ Title _____
 School/Company _____
 Dept _____
 Address _____
Give street address for UPS
 City & State _____ Zip _____

Applesoft Extensions

by Kerry Shetline

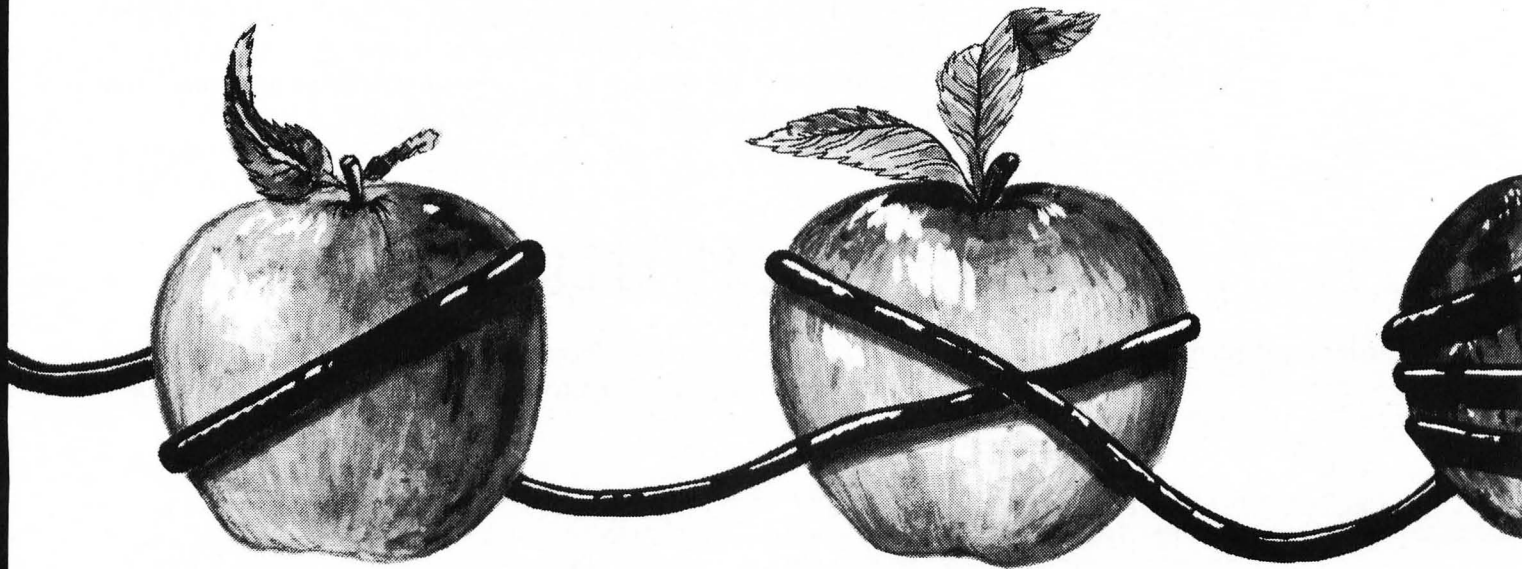
Applesoft Extensions is a group of Machine Language utilities requiring a 32K Apple™ with Applesoft and disk drive. It is included as the bonus program on this month's Apple™ Disk Version.

Applesoft Extensions is a set of Machine Language routines designed to add three new features to

HIMEM, and that HIMEM then be set to point to the memory just below the code. The following program lines will accomplish just that:

```
100 LINE = PEEK (116) * 256 + PEEK  
    (115): IF PEEK (LINE) < >  
    169 OR PEEK (LINE + 165) <  
    > 198 OR PEEK (LINE + 346)  
    < > 96 THEN LINE = LINE -  
347: HIMEM: LINE: PRINT CHR#  
(4)"BLDAD APLEXT.H,A"LINE
```

The line input statement allows you to input a string and have all of the characters, including quotes, commas, and colons, entered as part of that string. All ESC key cursor movements are available. The statement works by calling the monitor GETLN routine, and assigning a string to the contents of the input buffer. There is no removal of leading or trailing blanks, and no checking for delimiters. This is very useful for reading text files, and



Applesoft; a line input statement, a MID\$ assignment statement, and a statement similar to RESTORE which can set the DATA pointer to a specific line number. These features are accessed with the CALL statement, so they will not interfere with any "&" routines. The code is short, only 347 (decimal) bytes long, and relocatable, so that it can be placed just about anywhere in memory. For safety, I recommend that the code be loaded just below

```
110 ASSIGN = LINE + 69:RESTR = LI  
    NE + 295
```

To avoid lowering HIMEM every time the program is run, it checks for the presence of the extensions by testing a few bytes, then skips the loading procedure if the code is found. These lines should be placed at the very beginning of a program to avoid problems with resetting HIMEM.

eliminates the need for using file names in a program. The user can type a drive specification and not get the response "EXTRA IGNORED".

The syntax is: CALL LINE INPUT svar

LINE is a variable equal to the address at which the extensions code is located, and svar is any valid string variable. The loading routine



APPLE™ DV

listed above sets the variable LINE to the proper address (as well as the variables ASSIGN and RESTR used in the other extension statements). It should be noted that LINE INPUT generates no prompt.

The MID\$ assignment statement allows you to place a string directly into another string.

The syntax is: CALL ASSIGN MID\$(svar,expr)=sexpr

ASSIGN is a variable equal to the extensions' start address plus 69,

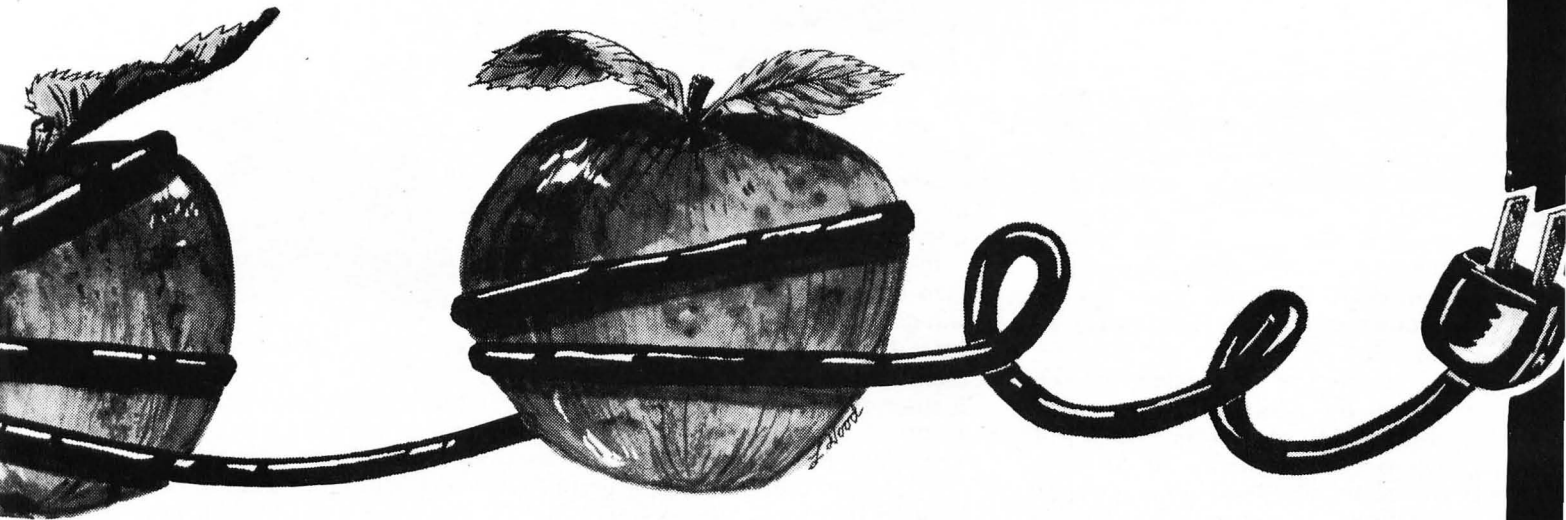
```
150 A$ = "AMPLE": CALL  
ASSIGN MID$(A$,2) = "P"  
160 B$ = "BACH": CALL  
ASSIGN MID$(B$,3) = "SIE"
```

In line 150, "M" is replaced by "P", and the result, A\$, contains the word "APPLE". In line 160, "CH" is replaced by "SI", and the remaining letter, "E", is placed at the end of B\$, so the resulting value of B\$ is "BASIE".

The last extension gives you the

The function of this statement is similar to that of the standard RESTORE statement, except that you can restore to any line containing a DATA statement, rather than only restoring to the very beginning of all the data.

Applesoft Extensions support normal Applesoft error handling. The "SYNTAX ERROR" message will be printed if any of the syntax requirements are violated. A "TYPE MISMATCH" error will



svar is any valid string variable, expr is a numerical expression, and sexpr is a string expression.

The string that will be changed is svar. Expr indicates the position in svar where sexpr will be placed. Expr may have any value from 1 to the length of svar. Sexpr can be any string expression so long as placing sexpr into svar does not result in a string that is longer than 255 characters. To clarify matters, let's look at some examples:

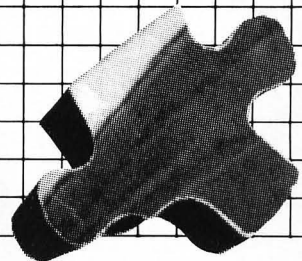
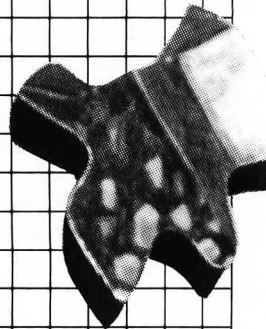
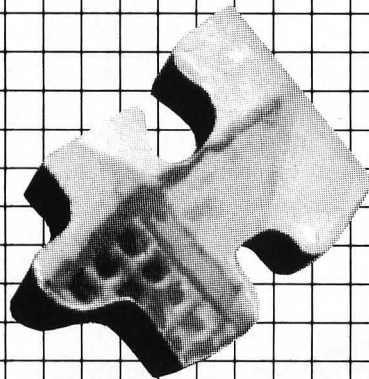
ability to restore the DATA pointer to any line number.

The syntax is: CALL RESTR GOTO linum

RESTR is a variable 295 greater than the starting address of the extensions code, and linum is the line number where the DATA pointer is to be set. The GOTO statement is required in the syntax so that renumbering utilities will properly change the line number reference in the statement.

occur if a numeric variable is used for svar, a string expression is used for expr, or a numeric expression is used for sexpr. An "ILLEGAL QUANTITY" error will result if expr is outside the valid range. "STRING TOO LONG" will be generated if the result of ASSIGN MID\$ is greater than 255 characters. Using a line number for linum that does not contain a DATA statement will cause an "OUT OF DATA" error.

PUZZLE



by Gary Cage

***Puzzle Jumble* is a graphics game for an Apple™ with Applesoft and 16K RAM.**

If you enjoy puzzles, you and your Apple are likely to spend many hours together with this game program. First a picture is drawn on the screen, using black, white, and flashing blocks. After you've studied the picture for a few moments, the computer scrambles it randomly, and you must try to reconstruct it in as few moves as possible.

The picture is created on the screen by printing 20 rows of blocks. Each row is stored in memory as a string of 36 text characters, which the computer then interprets as normal, inverse, or flashing blocks. The computer jumbles the picture by interchanging these strings, and you unjumble it by doing the same. All the rows are numbered on the screen, so you simply specify two rows to exchange, and the computer re-draws the picture accordingly.

As listed, the program contains the string assignment statements for five pictures in lines 8000-8995. This number can be easily changed by adding string assignment statements following these lines, and changing the ON-GOTO statement in line 4030 accordingly. The DIMENSIONS of MARK\$ and TRY in line 1000,

and the number 5 in the FOR-TO statement in line 6040 also need to be changed to reflect the new number of pictures.

In a future issue of *SoftSide* we plan to publish a program which will allow you to create and edit pictures for *Puzzle Jumble*, and save them as disk files. In the meantime, it's not difficult to create additional pictures of your own by hand. Here's the procedure:

1) Design your picture on graph paper. It should be exactly 20 squares high and 36 squares wide. You can use any combination of white, black, and flashing squares.

2) Type in 20 program lines, one for each horizontal row of blocks in your picture. If you want to be consistent with the format already in the program, begin the lines for picture number six at 9000 and number by tens, concluding with a RETURN statement at 9195. The structure of each string is the same:

a) The first character is a letter (A-T), indicating the proper order of the row in the picture. If several rows in sequence are the same, they can all have the same beginning let-

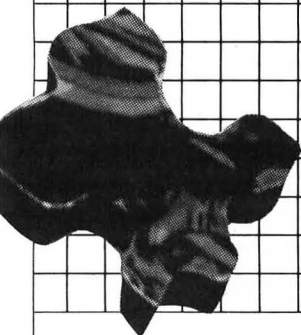
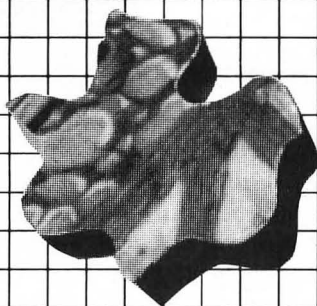
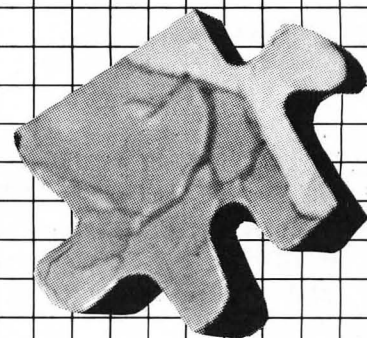
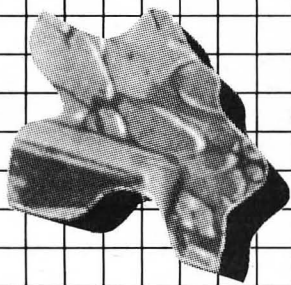
ter. (See, for example, lines 8060-8090.) If, however, you really want to be devious you can use rows (either in sequence or not) which are identical on the screen, but which do NOT have the same initial letters. These rows would then need to be put into their original order for the picture to be correctly solved.

b) Each of the other 36 characters indicates whether the corresponding block in the row is to be black (a normal space), white (inverse), or flashing. An F indicates flashing, a colon (:) indicates white, and any other character (such as an asterisk) indicates black.

Program Notes

There are a few PEEKs, POKEs, CALLs, and other useful techniques used in the coding which may be unfamiliar to some. In the initializa-

JUMBLE



screen window. Line 3110, for example, first reduces the width of the screen to 36 characters, and then moves the left margin four characters to the right. If you want to indent a number of lines of text, this is a much easier way to do it than using TAB or HTAB, or printing extra spaces with each line. The TEXT command restores the screen margins to their normal values.

The puzzle scrambling subroutine accomplishes its mission by taking each row of the puzzle and exchanging it with another row selected at random. This is repeated five times through with all 20 rows, and results in a thoroughly scrambled image.

Variables

ANS\$: Player's input.
 BELL\$: Equals CHR\$(7); causes speaker to beep when printed.
 CK: Flag to check if puzzle is in correct order (0 = yes, 1 = no).
 FRIE: Dummy variable used with FRE statement to clear unused strings from memory.
 H, H1, H2: Used for HTAB positions.

I, II: Loop variables.
 LOOP: Number of passes through the sound routine.
 MARK\$(A): Holds an asterisk for puzzle A if it was not completed.
 N1, N2: Puzzle line numbers to be exchanged.
 N1\$, N2\$: Input variables.
 NAMES\$: Author's name (for title routine).
 NOISE: Equals -16336, the memory address for clicking the speaker.
 NUM: Puzzle number.
 OLD: Holds old value of TST; used to see if previous line in puzzle is equal to current line.
 P1, P2: Puzzle lines to be exchanged in scrambling routine.
 PART: Part of the puzzle (1-20).
 PIC\$(A): Array of strings representing the puzzle.
 SOUND: Dummy variable used in speaker clicking loop.
 TIME: Delay loop variable.
 TITLES\$: Name of program, for title routine.
 TMP\$: Temporary holding variable; used for exchanging puzzle lines.
 TRY(A): Number of attempts to put puzzle A in order.
 TST: Holds ASCII value, minus 64, of the first character in each PIC\$ element; used in checking the correct picture order.
 V, V1, V2: Used for VTAB positions.

tion subroutine, BELL\$ is assigned a value of CHR\$(7), or CTRL-G; printing this string is then a convenient way of beeping the speaker. The variable NOISE is also used in generating sound; either PEEKing or POKEing this address will cause a click in the Apple™'s speaker.

In the title subroutine, a CALL -912 has the effect of scrolling the screen up one line, and a CALL -868 clears the current text line from the cursor to the right edge of the screen. These are handy monitor routines to remember.

In the instructions and picture display subroutines, the boundaries of the text window are manipulated in various ways to make screen formatting easier. As detailed in the Applesoft manual, POKEing memory addresses 32, 33, 34, and 35 changes the left margin, width, top margin, and bottom margin of the

```

#####
$ APPLESQT BASIC $
$ 'PUZZLE JUMBLE' $
$ AUTHOR: GARY CAGE $
$ (C) 1982 SOFTSIDE $
#####

```

```
10 GOTO 100
```

Subroutine to convert a string to inverse, flashing, and normal spaces, and display a line on the screen.

```

50 FOR I = 2 TO 37:M$ = MID$(P
  IC$(PART),I,1): IF M$ = ":" THEN
  INVERSE
60 IF M$ = "F" THEN FLASH
70 PRINT " "; NORMAL : NEXT : RETURN

```

Main program control section.

```
100 TEXT : HOME
```

Initialize variables.

```
110 GOSUB 1000
```

Print the title display.

```
120 GOSUB 2000
```

Print instructions.

```
130 GOSUB 3000
```

Present puzzles for unscrambling.

```
140 GOSUB 4000
```

Display the final scores.

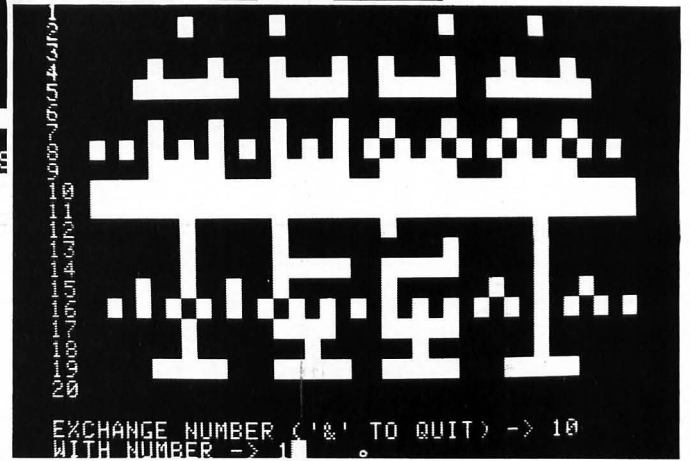
```
150 GOSUB 6000
160 VTAB 22: END
```

Initialization subroutine.

```

1000 DIM PIC$(20),MARK$(5),TRY(5
)
1010 BELL$ = CHR$(7)
1020 NAME$ = "BY G. CAGE"
1030 NOISE = - 16336
1040 TITLE$ = "PUZZLE JUMBLE"
1050 RETURN

```



Title subroutine.

```

2000 FOR I = 1 TO 12
2010 V = 12:H = 16
2020 V1 = I:V2 = 24 - I
2030 H1 = I + 4:H2 = 28 - I
2040 VTAB V: HTAB H1: PRINT TITL
  E$
2050 VTAB V1: HTAB H: PRINT TITL
  E$
2060 VTAB V: HTAB H2: PRINT TITL
  E$
2070 VTAB V2: HTAB H: PRINT TITL
  E$
2080 IF I = 1 THEN FOR TIME = 1
  TO 500: NEXT TIME
2090 IF I = 12 THEN GOTO 2120
2100 FOR TIME = 1 TO 50: NEXT TI
  ME
2110 HOME
2120 NEXT I
2130 FOR TIME = 1 TO 1000: NEXT
  TIME
2140 FOR I = 1 TO 5
2150 LOOP = 5: GOSUB 3500
2160 CALL - 912
2170 NEXT I
2180 VTAB 11: HTAB 20: PRINT "--
  -----"
2190 VTAB 13: HTAB 20: PRINT BEL
  L$;"-----"
2200 FOR TIME = 1 TO 200: NEXT T
  IME
2210 FOR I = 1 TO LEN (NAME$)
2220 VTAB 12: HTAB 41 - I: CALL
  - 868

```

```

2230 PRINT MID$(NAME$,1,I)
2240 LOOP = 5: GOSUB 3500
2250 NEXT I
2260 FOR I = 31 TO 20 STEP - 1
2270 VTAB 12: HTAB I: CALL - 86
  8
2280 PRINT NAME$
2290 LOOP = 5: GOSUB 3500
2300 NEXT I
2310 FOR TIME = 1 TO 1000: NEXT
  TIME
2320 FOR I = 1 TO 13
2330 LOOP = 1: GOSUB 3500
2340 CALL - 912
2350 NEXT I
2360 FOR TIME = 1 TO 500: NEXT T
  IME
2370 RETURN

```

Instructions subroutine.

```

3000 VTAB 8: PRINT "WOULD YOU LI
  KE INSTRUCTIONS? ";
3010 LOOP = 20: GOSUB 3500
3020 GET AN$
3030 IF AN$ = "N" THEN GOTO 324
  0
3040 IF AN$ < > "Y" THEN GOTO
  3020
3050 PRINT AN$
3060 HOME
3070 PRINT "INSTRUCTIONS:"
3080 PRINT "-----"
3090 FOR TIME = 1 TO 500: NEXT T
  IME

```

```

3100 LOOP = 20: GOSUB 3500
3110 POKE 33,36: POKE 32,4
3120 VTAB 5: PRINT "PUZZLE JUMB
LE' IS A GAME IN WHICH"
3130 PRINT "A PICTURE IS SHOWN T
O YOU. WHEN"
3140 PRINT "YOU PRESS 'B' TO BEG
IN, THE"
3150 PRINT "PICTURE IS JUMBLED,
FORMING A"
3160 PRINT "PUZZLE. YOU THEN HA
VE TO RE-"
3170 PRINT "CONSTRUCT THE PICTUR
E IN AS FEW"
3180 PRINT "MOVES AS POSSIBLE (P
RESSING '&'"
3190 PRINT "ALLOWS YOU TO SKIP T
O THE NEXT"
3200 PRINT "PICTURE)."
3210 VTAB 20: PRINT " PRESS 'SP
ACE BAR' TO BEGIN ";
3220 GET AN$
3230 IF AN$ < > CHR$(32) THEN
GOTO 3220
3240 PRINT AN$
3250 RETURN

```

Sound subroutine.

```

3500 FOR II = 1 TO LOOP
3510 SOUND = PEEK (NOISE) * PEEK
(NOISE)
3520 NEXT II
3530 RETURN

```

Main routine to display puzzles and decode them.

```

4000 TEXT : HOME
4010 FOR NUM = 1 TO 5
4020 TRY (NUM) = 0: MARK$(NUM) = ""
4030 ON NUM GOSUB 8000,8200,8400
,8600,8800
4040 GOSUB 5000
4050 HOME
4060 PRINT : HTAB 11: PRINT "PRE
SS 'B' TO BEGIN ";
4070 GET AN$
4080 IF AN$ < > "B" THEN GOTO
4070
4090 PRINT AN$
4100 TEXT : HOME
4110 GOSUB 5500
4120 GOSUB 5000
4130 CK = 0

```

```

4140 OLD = 0
4150 FOR PART = 1 TO 20
4160 TST = ASC (PIC$(PART)) - 64
4170 IF TST = OLD THEN GOTO 420
0
4180 IF TST < > PART THEN CK =
1:PART = 20
4190 OLD = TST
4200 NEXT PART
4210 IF CK = 0 THEN GOTO 4330
4220 HOME
4230 PRINT : INPUT "EXCHANGE NUM
BER ('&' TO QUIT) -> ";N1$
4240 IF N1$ = "&" THEN MARK$(NUM
) = "X": GOTO 4400
4250 N1 = INT ( VAL (N1$))
4260 IF N1 < 1 OR N1 > 20 THEN GOTO
4220
4270 VTAB 23: HTAB 1: CALL - 86
8: INPUT "WITH NUMBER -> ";N
2$
4280 N2 = INT ( VAL (N2$))
4290 IF N2 < 1 OR N2 > 20 THEN GOTO
4270
4300 TMP$ = PIC$(N1):PIC$(N1) = P
IC$(N2):PIC$(N2) = TMP$
4310 TRY (NUM) = TRY (NUM) + 1
4320 PART = N1: VTAB N1: HTAB 4: GOSUB
50:PART = N2: VTAB N2: HTAB
4: GOSUB 50: HOME : GOTO 413
0
4330 PRINT : PRINT BELL$;"YOU CO
MPLETED THIS PUZZLE IN ";TRY
(NUM);" TRIES."
4340 FOR TIME = 1 TO 2000: NEXT
TIME
4350 HOME
4360 PRINT : HTAB 10: PRINT "PRE
SS 'C' TO CONTINUE ";
4370 GET AN$
4380 IF AN$ < > "C" THEN GOTO
4370
4390 PRINT AN$
4400 TEXT : HOME
4410 FRIE = FRE (0)
4420 NEXT NUM
4430 RETURN

```

Subroutine to display a picture.

```

5000 TEXT : HOME
5010 FOR I = 1 TO 20
5020 PRINT I
5030 NEXT I

```

```

5040 POKE 33,36: POKE 32,3
5050 HOME
5060 FOR PART = 1 TO 20: GOSUB 5
0: NEXT
5070 POKE 32,0: POKE 33,40
5080 POKE 34,20
5090 RETURN

```

Subroutine to scramble the puzzle.

```

5500 VTAB 8: HTAB 10: PRINT "NOW
MIXING UP PICTURE"
5510 FOR I = 1 TO 5: FOR P1 = 1 TO
20
5520 P2 = INT ( RND (1) * 20) +
1
5530 TMP$ = PIC$(P1):PIC$(P1) = P
IC$(P2):PIC$(P2) = TMP$
5540 NEXT P1,I
5550 RETURN

```

Subroutine to display the scores.

```

6000 TEXT : HOME
6010 HTAB 14: PRINT "FINAL SCORE
S"
6020 HTAB 14: PRINT "-----
-"
6030 VTAB 6
6040 FOR I = 1 TO 5
6050 TRY$ = STR$(TRY(I))
6060 HTAB 5: PRINT "PUZZLE #";I;
SPC( 17 - LEN (TRY$));TRY$
;: IF MARK$(I) = "X" THEN PRINT
" TRIES ";MARK$(I): GOTO 608
0
6070 PRINT " TRIES"
6080 PRINT
6090 NEXT I
6100 VTAB 20: HTAB 5: PRINT "(X)
PUZZLE NOT COMPLETED"
6110 RETURN

```

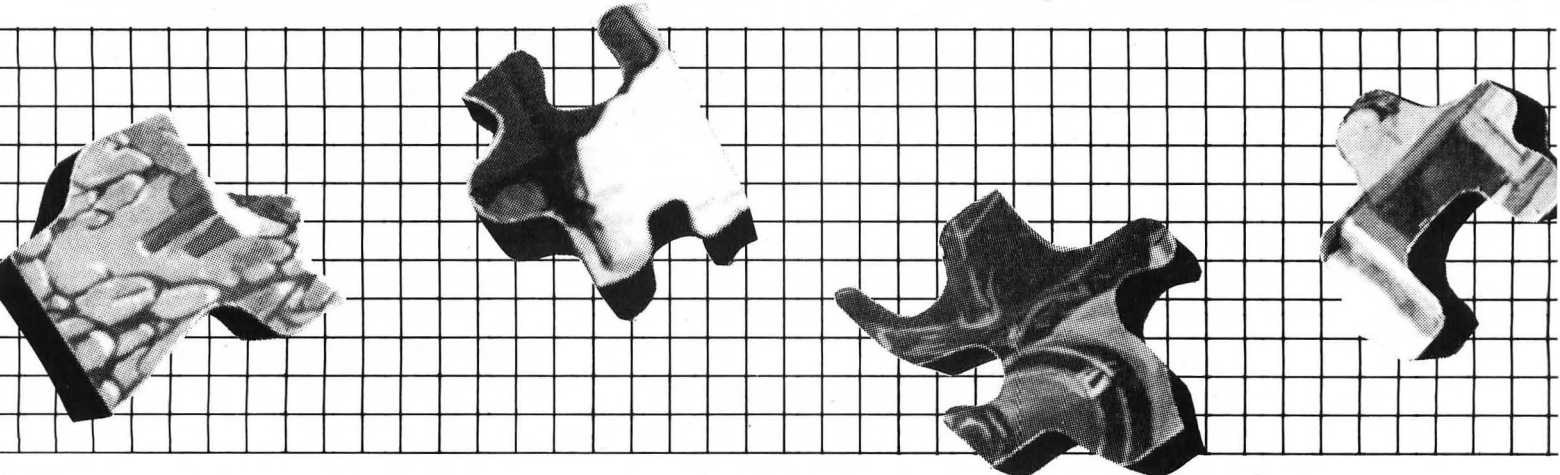
String assignment statements for defining the pictures.

Picture #1.

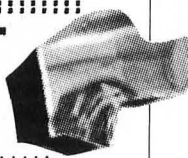
```

8000 PIC$(1) = "A:.....
:....."
8010 PIC$(2) = "B:#####:#####
#####:#####:"
8020 PIC$(3) = "C:#####:#####
#####:#####:"
8030 PIC$(4) = "D:#####:#####
#####:#####:"

```



8040 PIC\$(5) = "E:::~:::~:::~:::
 :~::~~::~::~~::~::~::"
 8050 PIC\$(6) = "F:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8060 PIC\$(7) = "G:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8070 PIC\$(8) = PIC\$(7)
 8080 PIC\$(9) = PIC\$(7)
 8090 PIC\$(10) = PIC\$(7)
 8100 PIC\$(11) = "K:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8110 PIC\$(12) = "L:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8120 PIC\$(13) = "M:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8130 PIC\$(14) = "N:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8140 PIC\$(15) = "O:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8150 PIC\$(16) = "P:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8160 PIC\$(17) = "Q:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8170 PIC\$(18) = "R:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8180 PIC\$(19) = "S:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8190 PIC\$(20) = "T:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8195 RETURN



Picture #2.



8200 PIC\$(1) = "A:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8210 PIC\$(2) = PIC\$(1)
 8220 PIC\$(3) = PIC\$(1)

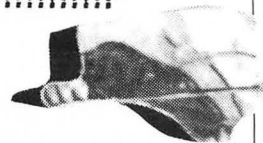
8230 PIC\$(4) = PIC\$(1)
 8240 PIC\$(5) = "E:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8250 PIC\$(6) = PIC\$(5)
 8260 PIC\$(7) = PIC\$(5)
 8270 PIC\$(8) = PIC\$(5)
 8280 PIC\$(9) = "I:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8290 PIC\$(10) = PIC\$(9)
 8300 PIC\$(11) = "K:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8310 PIC\$(12) = "L:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8320 PIC\$(13) = PIC\$(12)
 8330 PIC\$(14) = "N:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8340 PIC\$(15) = "O:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8350 PIC\$(16) = PIC\$(15)
 8360 PIC\$(17) = "Q:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8370 PIC\$(18) = "R:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8380 PIC\$(19) = PIC\$(18)
 8390 PIC\$(20) = "T:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8395 RETURN

Picture #3.

8400 PIC\$(1) = "A:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8410 PIC\$(2) = PIC\$(1)
 8420 PIC\$(3) = "C:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8430 PIC\$(4) = PIC\$(3)
 8440 PIC\$(5) = "E:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8450 PIC\$(6) = PIC\$(5)

8460 PIC\$(7) = "G:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8470 PIC\$(8) = "H:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8480 PIC\$(9) = "I:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8490 PIC\$(10) = "J:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8500 PIC\$(11) = PIC\$(10)
 8510 PIC\$(12) = "L:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8520 PIC\$(13) = PIC\$(12)
 8530 PIC\$(14) = "N:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"
 8540 PIC\$(15) = "O:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8550 PIC\$(16) = "P:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8560 PIC\$(17) = "Q:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8570 PIC\$(18) = "R:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8580 PIC\$(19) = "S:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8590 PIC\$(20) = "T:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8595 RETURN

Picture #4.



8600 PIC\$(1) = "A:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8610 PIC\$(2) = PIC\$(1)
 8620 PIC\$(3) = "C:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8630 PIC\$(4) = "D:::~:::~:::~::~::
 ~::~~::~::~~::~::~::"
 8640 PIC\$(5) = "E:::~:::~:::~::~::
 :~::~~::~::~~::~::~::"

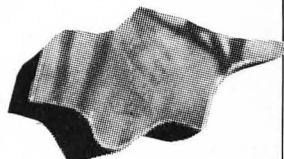
8650 PIC*(6) = PIC*(5)
 8660 PIC*(7) = "G*****F**
 *******"
 8670 PIC*(8) = "H*****
 :***"
 8680 PIC*(9) = "I*****
 t:*****"
 8690 PIC*(10) = "J*****
 t:*****"
 8700 PIC*(11) = "K*****
 :**"
 8710 PIC*(12) = "L*****
 :*****"
 8720 PIC*(13) = "M*****
 :*****"
 8730 PIC*(14) = PIC*(13)
 8740 PIC*(15) = PIC*(13)
 8750 PIC*(16) = PIC*(13)
 8760 PIC*(17) = "Q*****
 :*****"
 8770 PIC*(18) = "R*****
 *****"
 8780 PIC*(19) = "S*****
 :*****"
 8790 PIC*(20) = "T*****
 :*****"
 8795 RETURN

8900 PIC*(11) = "K*****
 *****"
 8910 PIC*(12) = "L*****
 *****"
 8920 PIC*(13) = "M*****
 *****"
 8930 PIC*(14) = "N*****
 *****"
 8940 PIC*(15) = "O*****
 *****"
 8950 PIC*(16) = "P*****
 *****"
 8960 PIC*(17) = "Q*****
 *****"
 8970 PIC*(18) = "R*****
 *****"
 8980 PIC*(19) = "S*****
 *****"
 8990 PIC*(20) = "T*****
 *****"
 8995 RETURN

APPLE™ SWAT TABLE FOR: PUZZLE JUMBLE

LINES	SWAT CODE	LENGTH
10 - 1000	NB	180
1010 - 2060	TH	207
2070 - 2180	VY	190
2190 - 2300	TX	202
2310 - 3040	RG	204
3050 - 3160	BC	308
3170 - 3520	IE	272
3530 - 4100	PO	186
4110 - 4220	EU	170
4230 - 4340	KO	410
4350 - 5020	IL	145
5030 - 5540	YV	221
5550 - 6100	RE	245
6110 - 8100	ZJ	484
8110 - 8200	NB	535
8210 - 8320	OJ	373
8330 - 8430	GD	459
8440 - 8550	NA	537
8560 - 8660	TK	518
8670 - 8780	IN	541
8790 - 8880	FI	527
8890 - 8980	ED	530
8990 - 8995	PV	59

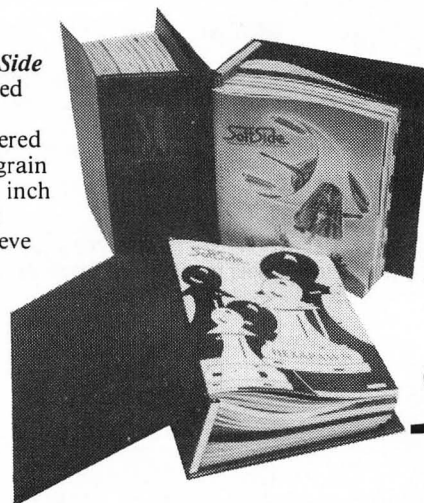
Picture #5.



8800 PIC*(1) = "A*****
 *****"
 8810 PIC*(2) = "B*****
 *****"
 8820 PIC*(3) = "C*****
 *****"
 8830 PIC*(4) = "D*****
 *****"
 8840 PIC*(5) = "E*****
 *****"
 8850 PIC*(6) = "F*****
 *****"
 8860 PIC*(7) = "G*****
 *****"
 8870 PIC*(8) = "H*****
 *****"
 8880 PIC*(9) = "I*****
 *****"
 8890 PIC*(10) = "J*****
 *****"

Protect Your Investment

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



you can label for easy identification. Each binder holds 12 issues.

8½ x 11..... \$7.95

Please include \$2.50 per order for shipping and handling.

See page 16 for ordering information & back issues bind-in card.

SoftSide

6 South Street
Millford NH 03055

Win \$500!

The First National Computer Owner Survey 50 Second Prizes of \$10 each!

In order to keep developing and bringing you very special hardware, software and publications, we've been commissioned to find out what you, the customer, wants and needs.

In addition to contributing to the computer owners' data base, you have a chance to win \$500... just for filling out this survey.

JUST TAKE A FEW MINUTES, ANSWER EVERY APPLICABLE QUESTION - YOU MUST TO BE ELIGIBLE - AND MAIL TO US NO LATER THAN OCTOBER 31, 1982. FOR 20¢ YOU COULD WIN HUNDREDS.

Entrants must be computer owners or users and answer every applicable question. A random drawing, eligibility approval and list of winners will be verified by a Notary Public. Winners will be notified by Dec. 31, 1982. Grand Prize winner gives IRV Brechner Enterprises the right to use name and photo in future surveys and advertising. No purchase necessary. Limit one entry per person. Entries must be postmarked no later than Oct. 31, 1982 and reach us by Nov. 15, 1982. Prizes include one cash award of \$500, and 50 cash prizes of \$10 each. All survey entries become property of IRV Brechner Enterprises; none will be returned. All prizes will be awarded by Nov. 31, 1982. All Federal, State and Local taxes are responsibility of the winner. This contest void where prohibited by law. For a prize winner list, send a self-addressed stamped envelope to IRV Brechner Enterprises, Box 264WOB, West Orange, N.J. 07052.

NAME _____ ADDRESS _____

CITY/STATE/ZIP _____ AGE _____ SEX _____ Circle: SINGLE MARRIED # CHILDREN _____

COMPUTERS(S) YOU OWN OR USE (Circle) APPLE ATARI TRS-80 IBM PC HEWLETT-PACKARD HEATH ZENITH OSBORNE

XEROX DEC TEXAS INST. NORTH STAR COMMODORE VECTOR CROMEMCO OTHER OWNED HOW LONG

DISK DRIVES _____ OWN HARD DISK? _____ OWN MODEM? _____ PRINTER BRAND _____ MONITOR BRAND _____

APPROX. # DISKETTES OWNED _____ BRAND PREFERENCE _____ OPERATING SYSTEM(S) _____

AMOUNT OF MEMORY (Circle) 8K 16K 24K 32K 48K 64K 128K MORE # DISKETTES PURCHASED/YEAR _____

LANGUAGES YOU PROGRAM WITH OR USE (Circle) BASIC FORTRAN COBOL MACHINE ASSEMBLER MONITORS

FORTH ALGOL PASCAL C ADA APL LISP CAI DO YOU WRITE YOUR OWN SOFTWARE (Circle) YES NO

APPROXIMATE NUMBER OF SOFTWARE PROGRAMS YOU OWN PER CATEGORY:

EDUCATION _____ BUSINESS _____ GAMES _____ SCIENTIFIC _____

HOBBY _____ HOME USE _____ OTHER _____

YOUR PROFESSION _____ PRIMARY USE FOR YOUR COMPUTER _____

APPROX ANNUAL INCOME (Optional) _____ MOST RECENT EDUCATION LEVEL (Circle) HIGH SCHOOL SOME COLLEGE

COLLEGE GRADUATE SOME GRADUATE SCHOOL MASTER'S DEGREE OTHER _____

WHICH PUBLICATIONS DO YOU SUBSCRIBE TO OR READ REGULARLY (Circle all that apply) APPLE ORCHARD BYTE CALL-APPLE

COMPUTE COMPUTERWORLD CREATIVE COMPUTING DESKTOP COMPUTING INTERFACE AGE INFOWORLD

MICROCOMPUTING MICRO MICROSYSTEMS NIBBLE PEELINGS II POPULAR COMPUTING PERSONAL COMPUTING

SOFTSIDE OTHERS _____

APPROX # COMPUTER BOOKS OWNED _____

WHAT NEW PRODUCTS, IDEAS, HARDWARE, SOFTWARE, PUBLICATIONS, ETC. DO YOU WISH TO SEE COME ABOUT?
Please be specific and use additional paper if necessary. Staple to survey when completed.

Signature _____

Mail all completed surveys by Oct. 31, 1982 to:
NATIONAL COMPUTER OWNERS' SURVEY • BOX 264WOB • WEST ORANGE, N.J. 07052

55

*Apple, Atari, TRS-80, IBM PC, Hewlett-Packard, Heath, Zenith, Osborne, Xerox DEC, Texas Inst., North Star, Commodore, Vector & Cromemco are all registered trademarks.

Quik Folio — A Small Investor's Small Portfolio Evaluator

by Rod Packer

This program was generously given to SoftSide by R. E. Packer, Ph.D. of Marana, AZ. Our thanks to Dr. Packer for allowing us to make the program available to our readers.

QuikFolio allows the amateur investor to fit polished and professional programming compactly onto his microcomputer screen. It's a learning tool, in barely two dozen easily understandable BASIC lines, that may be modified and used to sharpen programming skills, and add to investment skills for market profits.

*Readers interested in learning enough programming to customize their investing software may want to invest in the whole series, *The Computing Investor*, or *The Investor's Computer Handbook*. Both are available from *The Computing Investor*, 29 Estancia, Marana, AZ 85238.

The Need is to:

Start with clean screen
Allow more than BASIC's automatic 10-item list (array): 18 Quotes will fit down the screen..with 4 data for each stock (18 x 4).
Head the Report..
& give starting/ending guidance.

"Store-up" the 5 sub-headings to be used above each column.
Use each to head a column, as its Label..starting each heading on line 3..tabbing rightward to new column tab.
Display each heading..returning to underline it, on screen.
Reset Vertical Tab down one line for accepting entries.
For all "N" stocknames, in turn..tabbing downscreen a line for each.
Prepare to display at next tab-spot.
If "counter" V shows loops thru for final time..go to profit-figuring..return & repeat for next stockname.
When all figured..go do profit totals.
If stocknames already taken, then accept other inputs as numbers, indenting a space..and rejecting non-numeric.
On 1st time thru loop (V-counter = 0) accept & print stocknames' 1st eight letters onscreen (to column-width).
If no name or data is entered, then previous name was last item number. ..now accept Folio Cash..change it to a number..& leave data-taking loop.
Repeat until this column all filled in.
After a column is done..tab to next one & count up to next (Vth) column..
Go back to start its heading & entries.

Profits Module

Figure cost as shares held times costquote (1st & 2nd array items for each name)..give Cost only 2 decimals for \$\$.**
Figure Worth now as Shares times current quote..& make this figure \$\$.**
Accumulate costs & present worths into a growing folio total for each.
Figure net profit (worth minus cost)..show it for each stock before return.

Totals Display Module

Show, on bottom screenline (24)..Folio's total worth, including cash.
..Plus, at mid-bottom, Folio's net profit in dollars and cents.
..Plus, at right bottom, Percent gain (Folio Total Worth minus Total Cost) from cost..to 1 decimal place in %.
Hold this Report onscreen until investor gives an input to erase & start another.

Numeric-Input Check Module

If the entry has no numeric value..Go back and ask for re-entry.
If entry is numeric, move to next item.

Done in BASIC by:

```
1 HOME
5 DIM Q(18,4):N = 18:HT = 1

10 PRINT "QUIKFOLIO REPORT... RETURN KEY ENDS..
    OR 18TH ITEM"

95 DATA $STOCKS,$SHARES,BGT,BTE,QTE,NOW,$PROFIT
100 READ L$: VTAB 3: HTAB HT

101 PRINT L$: HTAB HT: PRINT "=====":
    VT = 4

110 FOR NAMES = 1 TO N:VT = VT + 1
111 HTAB HT: VTAB VT

115 IF V = 4 THEN GOSUB 150: NEXT NAME: GOTO 18
    0

119 IF V > 0 THEN HTAB HT + 1: INPUT "":Q$(NA
    ,V) = VAL (Q$): GOTO 1190
120 INPUT "":N$: VTAB VT: HTAB HT: PRINT LEFT$
    (N$,8) + "      "
125 IF N$ = "" THEN N = NAME - 1: INPUT "CASH BA
    LANCE $":C$:CASH = VAL (C$): GOTO 130

129 NEXT NAME
130 HT = HT + 8:V = V + 1

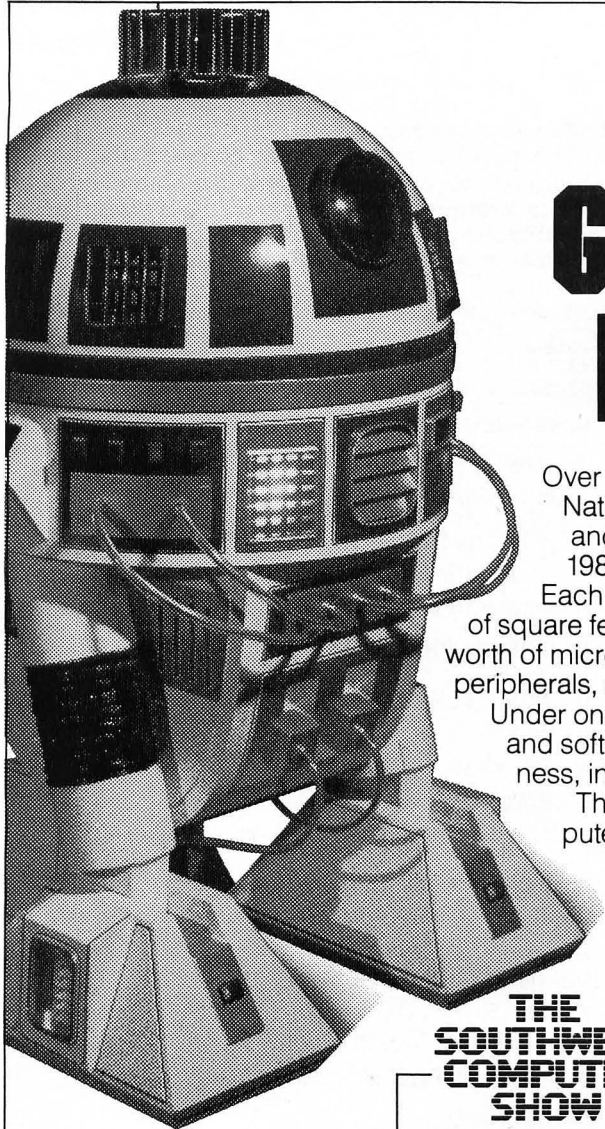
139 GOTO 100

150 C = Q(NA,1) * Q(NA,2)
151 C = INT (C * 100) / 100
155 W = Q(NA,1) * Q(NA,3)
156 W = INT (W * 100) / 100
160 TC = TC + C:TW = TW + W

179 P = W - C: PRINT P: RETURN

180 VTAB 24: HTAB 1: PRINT "TOTAL: $"TW + CASH;
185 HTAB 18: PRINT "NET: $" INT ((TW - TC) * 100
    ) / 100;
190 PC = (TW - TC) / TC:PC = INT (PC * 1000) / 1
    0: HTAB 34: PRINT PC"%";
199 HTAB 40: GET A$: RUN

1190 IF VAL (Q$) = 0 THEN 111
1191 GOTO 129
```



HAVE WE GOT A PROGRAM FOR YOU IN '82

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

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

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

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

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

THE NATIONAL COMPUTER SHOWS

Ticket Information

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

THE SOUTHWEST COMPUTER SHOW

Dallas
Dallas Market Hall

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

DIRECTIONS:
2200 STEMMONS FREEWAY
(AT INDUSTRIAL BLVD)

THE NEW YORK COMPUTER SHOW

Uniondale, Long Island
Nassau Coliseum

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

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

THE TWIN CITIES COMPUTER SHOW

Minneapolis
Minn. Auditorium
& Convention Hall
Third Avenue

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

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

THE MID-ATLANTIC COMPUTER SHOW

Washington, DC
DC Armory/Starplex
Across from RFK Stadium

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

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

THE MID-WEST COMPUTER SHOW

Chicago
(Arlington Heights)
Arlington Park Racetrack
Exhibition Center

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

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

THE NORTHEAST COMPUTER SHOW

Boston
Hynes Auditorium/
Prudential Center

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

DIRECTIONS: TAKE MASS
PIKE TO PRUDENTIAL
CENTER EXIT

THE SOUTHEAST COMPUTER SHOW

Atlanta
Atlanta Civic Center

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

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

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

Firebird

A review by Hartley G. Lesser

from Gebelli Software. System requirements: 48K Apple II or Apple II Plus with one disk drive — thirteen or sixteen sector controller. Suggested retail price: \$29.95.

Nasir Gebelli's genius, responsible for the creation of *Space Eggs*, *Autobahn*, *Cyber Strike*, and *Gorgon* (to name but a few!), bade farewell to Sirius Software not long ago. A new software company was to be formed. High expectations, based on past performance, awaited the company's initial software release. Would the game measure up to the high standards synonymous with the name Nasir Gebelli?

The introduction of *Firebird* from Gebelli Software, an arcade-style game, does not have the dynamic difference one might have expected, save for the game's firefighting theme. Nonetheless, the game is very exciting, and requires more than a modicum of dexterity plus the knowledge of several phrases containing explicit utterances symbolic of total frustration.

The package cover brightly illustrates the game's hero, "Piggo the Fireman." An iron-on transfer will be found inside, suitable for consignment to a T-shirt or other fine garment worthy of this game's logo. *Firebird* boots directly on either thirteen or sixteen sector disks, and the player is offered a chance to observe a demonstration game controlled by the computer, the outcome of which is definitely prophetic.

The initial graphic representation depicts a seventy-two room building. There are eight stories, nine rooms on each level. "Piggo" appears to be nothing more than one's basic oinker, save for his fireman's hat, fire-jacket, and asbestos gloves.

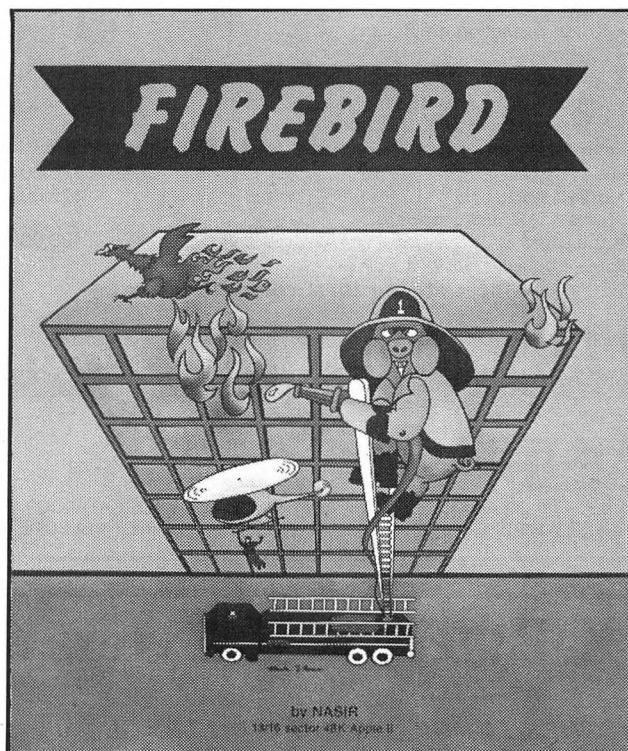
"Piggo" begins the session loitering at the base of a mobile, eight-story high ladder. A helicopter chatters noisily above the building, flying back and forth from screen edge to screen edge.

All seems rather peaceful, initially — merely the calm before the firestorm, as the demonstration program has so aptly demonstrated. A single page of instructions supplied with the disk clearly indicates that the player's responsibilities are two-fold.

First, "Piggo" must strive to prevent the building from burning down. Secondly, the helpless occupants of the edifice must be rescued as they leap from their burning rooms.

The arsonist, yet to make its appearance, is something called a "Firebird." The picture on the software package displays something akin to a wild turkey cavorting across the building's rooftop. A fiery wake is dispensed from the bird'sash-end...which causes the fires to begin blazing. An annoyed pig, attired in a firefighting suit, stands atop a fire ladder, looking most perplexed.

The game commences on the press of any key. Within seconds, the "Firebird" appears, randomly dropping fire



on the rooms of the building's upper level. Initial concern turns to relief as the gamer notes how easily these first few fires are extinguished. However, as the game progresses, the bird's speed increases, as do the number of fires he's spreading.

The player manipulates "Piggo" to the burning rooms, where the firepig sprays water from the firehose, attempting to douse the flames. But, the fires are not limited to a single level. Once a room is gutted by fire, the space the room occupied is eliminated from the building. Should flames drop onto the destroyed section, the next lowest level catches on fire!

"Piggo" is constantly running up and down the ladder, trying to extinguish flames on several different levels at once. All the time, additional fires are being started.

There is more — another responsibility is to try and accomplish rescue of the residents. "Piggo" must be positioned to the left side of a burning room in order to catch the poor unfortunate leaping to his or her death from the blazing apartment. "Piggo," after seizing the victim, must then climb the ladder and deposit his passenger upon the uppermost rung. The helicopter will then, and only then, pick up the salvaged tenant on its pass overhead. If this is accomplished, the chopper will deposit an undamaged room onto the building during its return flight, in an area where it is needed the most.



THE



BIG MATH ATTACK™

Challenging new math program . . .

Sharpen your skills by entering the correct answer before the equation 'lands' on your city! Provides hours of educational entertainment.

Features:

- Full color
- High resolution graphics
- Animation
- Sound
- Four math functions (+, -, x, ÷)
- Two levels for each function



Recommended for grades 1-6. Available for the ATARI & APPLE II.

ATARI 16K (cass.) \$20.00

ATARI 24K (disk) \$25.00

Requires ATARI BASIC cartridge

APPLE II (disk) DOS 3.2/3.3 \$25.00

Requires Applesoft Basic in ROM.

Ask for it at your local computer store.



or order direct:

H.E.S.I.S.

P.O. Box 147
Garden City, MI 48135
(313) 595-4722

Please add:

\$1.50 shipping/handling
\$1.50 C.O.D.

WRITE FOR FREE CATALOG
DEALER INQUIRIES WELCOME

• K-Byters • 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

APPLE™

Should our brave firepig rescue someone and not take him to the top of the ladder, the player will find it impossible to do further combat with the flames. There is also the possibility of rescuing a second tenant while another one clings to the top rung of the ladder, awaiting the helicopter. This will totally negate "Piggo's" fire fighting abilities, for only one rescue pickup may occur at one time. Poor "Piggo" must wait with his latest save until the ladder is clear, watching the fire eat away at the precious rooms around him.

Danger abounds, even during a rescue attempt. If "Piggo" is standing directly over a flaming apartment when a resident jumps from the room, he will be knocked from the ladder and will join the panic-stricken tenant in a free-fall to the ground. The player has a total of only three firepigs for the game, so care must be taken in this regard.

"The game presents an immense challenge, not only for point accrual, but in learning to maintain and douse one's temper while battling the fires."

"Piggo" is manipulated through the use of the computer keyboard. The "A" and "Z" keys control movement up and down the ladder, while the left and right arrows direct the ladder in corresponding directions. For dousing flames, the space bar should be pressed, sending a spray of water from the firehose onto the fire.

Should these particular keys not meet with the player's approval, Gebelli has allowed for key redefinition. CTRL-C enters this change mode, and the player merely types in the keypress desired for the appropriate action.

The game also includes a fantastic function that will save the gamer numerous migraines. What has a player had to do in the past when the phone rings, or the doorbell announces a visitor, during a hotly contested competition? Why, angrily turn off the computer. No longer, for Mr. Gebelli has provided an ESC function that holds the game indefinitely for the player at the position play was halted. A simple press of the ESC key and all activity is frozen and saved, awaiting your return.

The conclusion of the contest occurs when all three firepigs have been lost, or when two or more columns of rooms have burned to the ground. If the gamer is able to save the bottom level of apartments in an undamaged condition, 5000 bonus points are earned, and the game is restarted with your score intact. Using a CTRL-R will also restart the game at any point, but all accumulated points will be lost.

Points are earned for the following accomplishments:

- 1) Quenching a room fire: 25 points
- 2) Saving a falling victim: 50 points
- 3) Helicopter rescue: 100 points

The player's reward comes in an improvement of his previous score, each time the game is played — no easy matter at all. The contest is frustrating, for the chance of defeating the "Firebird" is basically nil. The game presents an immense challenge, not only for point accrual, but in learning to maintain and douse one's temper while battling the fires. This game is a worthwhile addition to any computer gamer's software library.



**IN CASE
YOU DIDN'T KNOW...**



Z.E.S. - THE ULTIMATE IN COMPUTER AIDED INSTRUCTION IS FOR APPLE* COMPUTERS ONLY!

Avant-Garde Creations announces the Z.E.S. system of computer aided instruction from Zenith Coaching in Australia.

Z.E.S. is the ultimate educator's or businessperson's tool for creating lessons in any subject **WITH NO PROGRAMMING KNOWLEDGE NECESSARY.**

Z.E.S. is **NOT** a language, but a completely menu driven system that features:

- full lesson creation/amendment facilities
- high resolution graphics and animation in color
- cartesian graphs
- full error checking, field delimiters and prompting
- provisions for hints in each question
- branching to revision questions (up to 9 revision questions for each normal question)
- **ABSOLUTELY NO PROGRAMMING KNOWLEDGE REQUIRED**
- elaborate student record keeping including:
 1. the student's general status on the lesson
 2. summary of the student's performance
 3. detailed report
 4. student's answers
 5. class report

The Z.E.S. System consists of 4 disks, the PROGRAM disk, the GRAPHICS disk, the MODULE disk and the MODULE FORMAT/COPY disk, plus a 130-page manual in deluxe ring binder. System Price: \$250, Demonstration Pkg: \$10

13 Z.E.S. COURSEWARE MODULES (MAY BE USED ALONE OR IN CONJUNCTION WITH THE COMPLETE Z.E.S. AUTHORIZING SYSTEM.) **29.95 EACH**

- | | |
|-----------------------------------|-----------------------------------|
| 1. Phrases and Clauses | 8. The Quadratic and the Parabola |
| 2. Poetry | 9. Statistics |
| 3. Introduction to Weather Charts | 10. The Heart |
| 4. Mechanics and Motion | 11. The Digestive System |
| 5. Electricity | 12. Gas Laws |
| 6. Functions | 13. The Definite Article |
| 7. The Real Number System | |

YOU CAN EARN \$\$\$\$ CREATING NEW CLASSROOM COURSEWARE. Suitable modules created on the ZES Authoring System can be entered into our resource bank. Teacher-authors receive royalties on any courseware they have written. Call for details.

For your Apple*. All systems and courseware require: Applesoft* in ROM, 48K, disk, DOS 3.2 or 3.3

Z.E.S. Authoring System **\$250**
 Demonstration Disk **\$10**
 COURSEWARE MODULES: **\$29.95 each**

Ask your dealer or order direct from:

Avant-Garde Creations,
PO Box 30160 Eugene, OR 97403
(503) 345-3043



Visa/Mastercard accepted



* Apple is a registered trademark of Apple Computer, Inc.



by W. Morris & J. Cope

Tunein is a music/memory game program for a 16K ATARI® 400/800 with ATARI® BASIC cartridge and one joystick.

Tunein is a one to four player game which tests the player's ability to store and recall an increasingly lengthy series of musical notes; it has been designed to be played by people of all ages. In writing *Tunein*, we have attempted to illustrate how "education" and "fun" do not have to be mutually exclusive concepts. Properly designed software in this field should allow for both intellectual and emotional satisfaction without being too obvious in realizing either goal.

Another of our intentions in writing this program was to illustrate, within a program setting, how the inherent power of the ATARI® can overcome apparent limitations of the microcomputer. "Limited to only four colors on the screen," you say? Well, *Tunein* features eight rectangular boxes of eight different colors! The careful use of a redesigned character set and player/missile graphics made short shrift of that difficulty. Similarly, the giant cursor, which is central to the game, is not done with any *Machine Language* programming but instead, uses some redesigned characters to accomplish the desired effect. Let no one kid you, the ATARI® is **powerful!**

Game Instructions

Tunein is a relatively straightforward game to learn how to play. If you are familiar with the popular children's game of "Simon Says," you already know most of the game's procedure. If not, read on....

Each game will commence by placing eight colored squares about the program's title. Each square represents one note of a musical octave. The ATARI® will cycle a giant cursor through each of these squares until the player(s) are ready to play. The first task facing the player(s) at this point is to set the initial playing conditions:

1. Use the SELECT key to indicate the number of players (1-4). You will know the number of players for each game by the presence of the initial scores in each corner.
2. Once you have selected the number of players, use the START key to commence play.
3. After completing one successful turn, the OPTION key will allow you to increase the length of the series of notes you will have to repeat to complete the current level and increase your score. The game is set to proceed in increments of one note (ten points). If, at any time,

you wish to start another game, use the START key to terminate play.

Each player has three "lives" during each game. Just prior to your turn, the score display will change to indicate the number of lives you have left (in asterisks) and the number of notes to be repeated in order to score.

To start your turn, press the joystick button. The ATARI® will flash the program title and play the sequence of notes while placing the giant "cursor" over one of the eight squares. The program title will disappear and wait for you to press the joystick button to indicate your readiness to play. Simply use the joystick to move the cursor over the appropriate square. Once this has been accomplished, press the joystick button to finalize your choice. A correct choice will result in the note being played again. An incorrect choice will elicit an appropriate "raspberry" from your ATARI®.

Each successful turn will result in one note being added to the player's sequence on the next turn (up to a maximum of 100 notes at the first level). After the 10th successful guess, the increase is by steps of 2 until you reach 20 after which it increases by 3 until 30 successive notes can be repeated. The increases follow a similar pattern until you reach 100 successful notes. (If you can repeat a 100 note pattern, please

contact the Guinness Book of World Records! Our best record is somewhat lower...)

As mentioned earlier, using the option key at the beginning of each turn will allow you to increase the length of the sequence you must repeat, up to the maximum allowable for that level. The game has been set to have a "wrap around" effect should you try to exceed the maximum level.

An unsuccessful turn will result in a repeat of the same level of difficulty on the next turn unless you have exhausted your three lives. Should this, in fact, be the case, you are eliminated from the game. *Tunein* will continue until all of the players have eliminated themselves. It will then commence a new game.

Variables

J, K: Coordinates for the program "cursor".

LF: The number of lives for each player.

LV: The level of the game (1-4 player game).

M: One of eight possible notes to be played.

MM: The number of notes that have been repeated by the player during his or her current turn.

N: An array containing the frequency of one of the eight possible notes used in the program.

NO: The maximum number of notes to be played at the current level of the program.

NN: The number of notes to be played at each stage during a player's turn.

P: Player counter.

PM: Player/Missile base address.

S1, S2: Coordinate points to print each player's score.

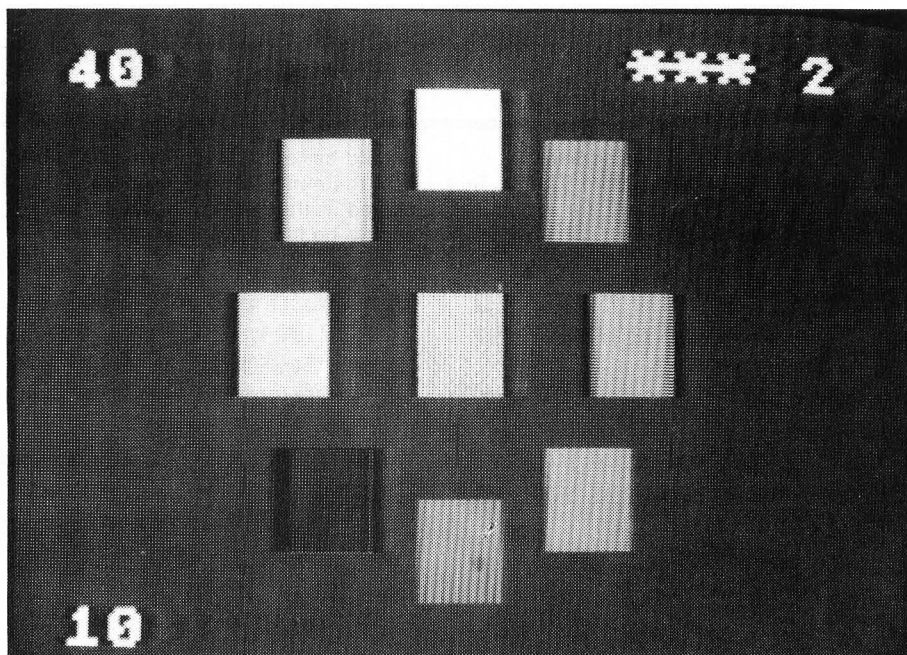
SC: Score for each player.

UU: Base address of redefined character set.

W: Variable set by the joystick to indicate the array element of J and K to help position the "cursor."

X\$: A "dummy" string used to store the Machine Language routine used to redefine the character set.

Y\$: Another "dummy" string used to store the Machine Language routine used to clear out the player-missile area in memory. All other variables are used as counters.



```

#####
$      ATARI BASIC      $
$      "TUNEIN"        $
$  AUTHORS: WILLIAM MORRIS $
$      AND JOHN COPE    $
$  (C) 1982  SOFTSIDE  $
#####
    
```

After clearing the screen and inhibiting the cursor, display the title page.

```

3 ? " ":POKE 752,1:SETCOLOR 4,3,0:SETC
OLOR 2,3,0:POSITION 17,8:? "TUNE IN"
4 POSITION 6,18:? "(c) Wm. Morris & J.
  Cope 1982";
    
```

All of the initial variable values are set.

```

10 DIM N(100),M(8),LV(4),NO(4),SC(4),L
  F(4),J(8),K(8),X$(32),Y$(24)
11 M(1)=60:M(2)=64:M(3)=72:M(4)=81:M(5
  )=91:M(6)=96:M(7)=108:M(8)=121
12 J(1)=9:J(2)=12:J(3)=13:J(4)=12:J(5)
  =9:J(6)=6:J(7)=5:J(8)=6
13 K(1)=1:K(2)=2:K(3)=5:K(4)=8:K(5)=9:
  K(6)=8:K(7)=5:K(8)=2
    
```

After reserving memory space for the redefined character set and using UU to store the address of the top of memory, a full screen graphics 2 mode is set and the cursor is inhibited.

```

20 POKE 106,PEEK(106)-4:UU=256#PEEK(10
  6):GOSUB 20000:GRAPHICS 18:POKE 752,1
    
```

Set all of the color registers to a uniform value while other matters are attended to prior to start of play.

```

30 FOR Z=704 TO 712:POKE Z,48:NEXT Z
    
```

After moving the ROM based character set into RAM in line 40, key characters are redefined to form the basis of the flashing cursor in the following line.

```

40 U=USR(ADR(X$),57344,UU):POKE 756,UU
  /256
50 FOR Z=456+UU TO 463+UU STEP 2:POKE
  Z,85:POKE Z+1,170:POKE Z+8,255:POKE Z+
  9,255:POKE Z-8,170:POKE Z-7,85:NEXT Z
    
```

A regular playfield is selected and memory is reserved for player-missile graphics below our redefined character set. The page address of player/missile graphics address is POKEd into location 54279, shadow register 623 is set to give the redefined cursor priority over the player/missile graphic figures and their base address is calculated in the variable PM.

```

60 POKE 559,62:Z=PEEK(106)-12:POKE 542
  79,Z:POKE 623,4:PM=Z#256
    
```

After using the Machine Language routine stored at the address of Y\$ to clear out the player missile area in memory, line 80 sets the shape of each of the four players to be the same as the redefined rectangular characters. The size of each of the players is set to the same size (locations 53256-53259) and their

horizontal positions on the screen are set (locations 53248-53251). The players are then "turned on" for display (location 53277).

```
70 U=USR(ADR(Y#),PM)
80 X=PM+1024:FOR Z=64 TO 95:POKE X+Z,2
55:POKE X+Z+256,255:POKE X+Z+608,255:POKE X+Z+864,255:NEXT Z
90 FOR Z=53256 TO 53259:POKE Z,1:NEXT Z
100 POKE 53248,96:POKE 53249,144:POKE 53250,96:POKE 53251,144:POKE 53277,3:GOSUB 110:GOTO 200
```

Subroutine to position the redefined rectangles on the screen.

```
110 POSITION 9,1:? #6;"ZZ";:POSITION 9,2:? #6;"ZZ";
120 POSITION 5,6:? #6;"ZZ";:POSITION 5,5:? #6;"ZZ";
130 POSITION 13,6:? #6;"zz";:POSITION 13,5:? #6;"zz";
140 POSITION 9,9:? #6;"zz";:POSITION 9,10:? #6;"zz";:RETURN
```

The game screen display is set with line 210 setting all of the SETCOLOR and player/missile color registers. Line 215 checks memory location 53279 to see if any of the console keys have been pressed or if the joystick button has been pressed. If not, the program loops continually through this line. Line 230 registers the SELECT button being pressed.

```
200 PL=3:GOSUB 400:GOSUB 500
210 POKE 704,118:POKE 705,84:POKE 706,178:POKE 707,70:POKE 708,28:POKE 709,132:POKE 710,250:POKE 711,52
215 GOSUB 10000:IF PEEK(53279)=7 AND 5 TRIG(0)<>0 THEN 215
220 GOSUB 300:IF STRIG(0)=0 THEN GOSUB 500:GOTO 1000
230 IF PEEK(53279)=5 THEN GOSUB 400
240 GOTO 220
```

The redefined characters are first printed in line 300 and then printed in reverse order in line 310 which gives the illusion of a "flashing" checkerboard cursor. The remaining lines in this section check the console registers to see if the game

should be terminated (lines 320-330) or if the number of notes to be played at that level of difficulty should be changed (line 340).

```
300 POSITION 8,5:? #6;" YY ";:POSITION 9,6:? #6;"YY":FOR U=1 TO 30:NEXT U
310 POSITION 9,6:? #6;"XX";:POSITION 9,5:? #6;"XX":FOR U=1 TO 30:NEXT U
320 IF PEEK(53279)=6 THEN POP:GOTO 200
330 IF PEEK(53279)=3 AND P=0 THEN POP:GOTO 200
340 IF PEEK(53279)=3 THEN NO(P)=NO(P)+1:POP:GOTO 1055
350 RETURN
400 PL=PL+1:IF PL=5 THEN PL=1
```

The number of players is registered and their initial scores are displayed.

```
410 POSITION 1,0:? #6;"00000 ";:POSITION 14,0:? #6;" ";:POSITION 1,11:? #6;" ";:POSITION 14,11:? #6;" ";
420 ON PL GOTO 460,450,440,430
430 POSITION 14,11:? #6;"00000";
440 POSITION 1,11:? #6;"00000";
450 POSITION 14,0:? #6;"00000";
460 RETURN
```

The initial parameters for the game are set.

```
500 FOR P=1 TO 4:LV(P)=1:SC(P)=0:LF(P)=3:NO(P)=1:NEXT P:P=0:RETURN
```

This line marks the initiation of the actual game sequence. The variable Y counts the number of players. If it is equal to zero, the program exits this segment.

```
1000 Y=0:FOR Z=1 TO PL:Y=Y+LF(Z):NEXT Z:IF Y=0 THEN 9000
1010 P=P+1:IF P>PL THEN P=1
1020 IF LF(P)=0 THEN 1010
1030 FOR Z=1 TO LV(P)*10:N(Z)=INT(RND(0)*8+1):NEXT Z
1040 S2=0:IF P>2 THEN S2=11
1050 S1=1:IF P=2 OR P=4 THEN S1=14
1055 IF NO(P)>10 THEN NO(P)=1
```

The previous score for the current player is wiped out and the number of lives left in this game are

displayed along with the number of notes to be played in this round.

```
1060 POSITION S1,S2:? #6;" ";:POSITION S1,S2:FOR Z=1 TO LF(P)? #6;"*";:NEXT Z:? #6;" ";(LV(P)-1)*10+NO(P);
```

The accompanying sound to the display in line 1060 is set.

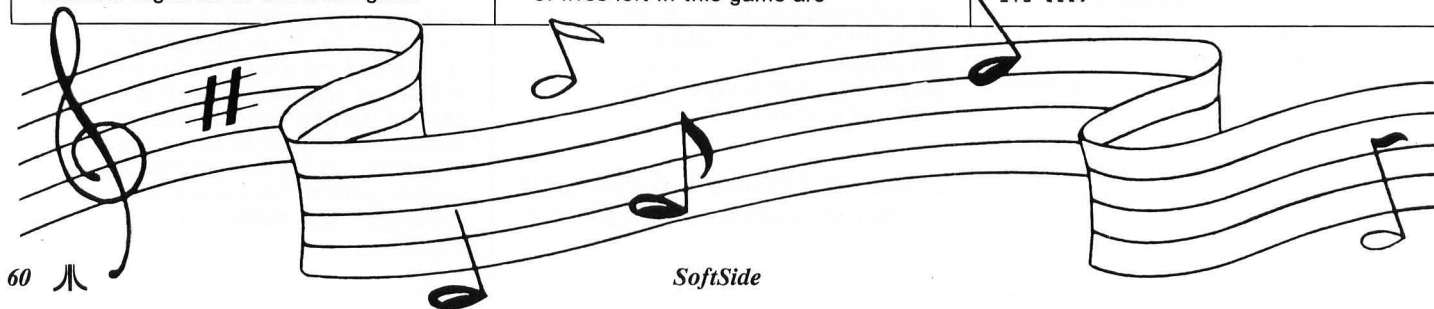
```
1070 FOR UZ=15 TO 0 STEP -.5:FOR UY=3 TO 0 STEP -.1:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT UZ
```

This segment plays the number of notes contained in the variable NN and places the cursor over the appropriate redefined character or player/missile rectangle. The subroutine in line 110 is accessed after the cursor has been erased to reprint the redefined character. This procedure is not necessary if the cursor has been printed over a player/missile character as the priority register has been set to display the player if no character is covering it.

```
1100 GOSUB 300:NN=0:IF STRIG(0)<>0 THEN N 1100
1110 GOSUB 2000:NN=NN+LV(P):FOR Z=1 TO 4:GOSUB 300:NEXT Z:IF NN>LV(P)*NO(P) THEN 3000
1120 FOR Z=1 TO NN:SOUND 0,M(N(Z)),10,B
1130 POSITION 9,6:? #6;"IN";:POSITION 8,5:? #6;"TUNE";
1140 POSITION J(N(Z)),K(N(Z))? #6;"YY";:POSITION J(N(Z)),K(N(Z))+1:? #6;"YY";
1150 FOR Y=1 TO 100:NEXT Y:POSITION J(N(Z)),K(N(Z))? #6;" ";:POSITION J(N(Z)),K(N(Z))+1:? #6;" ";:GOSUB 110
1160 SOUND 0,0,0,0:FOR Y=1 TO 10:NEXT Y:NEXT Z:MM=0
```

Player input is handled within this section. Note how line 1200 sets memory location 77 to 0 in order to prevent the attract flag from being set. MM, the number of notes input by the player, is compared to the number of notes actually played. If it is equal, the program exits this segment.

```
1200 POKE 77,0:MM=MM+1:IF MM>NN THEN GOTO 1110
```



```
1210 IF STICK(0)=15 THEN GOSUB 300:GOTO 1210
```

These lines register the movement of the joystick and set the variable W to the appropriate number to position the giant cursor.

```
1220 WW=STICK(0):IF STICK(0)=14 THEN W=1
1230 IF STICK(0)=6 THEN W=2
1240 IF STICK(0)=7 THEN W=3
1250 IF STICK(0)=5 THEN W=4
1260 IF STICK(0)=13 THEN W=5
1270 IF STICK(0)=9 THEN W=6
1280 IF STICK(0)=11 THEN W=7
1290 IF STICK(0)=10 THEN W=8
```

After wiping out the checkerboard cursor at the center of the screen in line 13000, it is printed in the following line at the new location indicated by the player. The following two lines read the joystick to see if the button has been pressed to register a choice. Line 1340 then erases the cursor from the new location and reprints the erased character by accessing the subroutine at line 110.

```
1300 POSITION 9,6:?" #6;" ";;POSITION 9,5:?" #6;" "
1310 POSITION J(W),K(W):?" #6;"YY";POSITION J(W),K(W)+1:?" #6;"YY";
1320 IF STRIG(0)=0 THEN 1400
1330 IF STICK(0)=WW THEN 1320
1340 POSITION J(W),K(W):?" #6;" ";;POSITION J(W),K(W)+1:?" #6;" ";;GOSUB 110:GOTO 1210
```

This series of lines constitute the evaluation section of the program. Line 1400 branches to 1500 if the guess is correct. Line 1410 registers an audible indicator of the incorrect guess and decreases the number of "lives" left to the player. The reprinting of the rectangle is repeated as outlined earlier. The player's successful guess is acknowledged in 1500 by replaying the correct note and incrementing his score.

```
1400 IF W=N(MM) THEN 1500
```

```
1410 SOUND 0,255,10,4:FOR UZ=1 TO 100:NEXT UZ:SOUND 0,0,0,0:LF(P)=LF(P)-1
```

```
1420 POSITION J(W),K(W):?" #6;" ";;POSITION J(W),K(W)+1:?" #6;" ";;GOSUB 110:GOTO 1000
```

```
1500 SOUND 0,M(W),10,8:FOR Z=1 TO 80:NEXT Z:SC(P)=SC(P)+10*LV(P):GOSUB 2000
1510 POSITION J(W),K(W):?" #6;" ";;POSITION J(W),K(W)+1:?" #6;" ";;GOSUB 110:SOUND 0,0,0,0
1520 FOR Z=1 TO 25:NEXT Z:GOTO 1200
```

Print the current player's score.

```
2000 POSITION 51,52:?" #6;" ";;POSITION 51,52:?" #6;"SC(P);:RETURN
```

The player's successful passage through the current level of difficulty results in an upgrading of both the number of notes to be played and the level of difficulty in lines 3000-3010.

```
3000 FOR UZ=16 TO 0 STEP -2:FOR UY=0 TO 5:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT UZ:NO(P)=NO(P)+1
3010 IF NO(P)>10 THEN NO(P)=1:LV(P)=LV(P)+1:IF LV(P)>9 THEN 10000
3020 GOTO 1000
```

The "end game" portion of Tunein. If the console button has not been pressed and the joystick button has not been struck, the program will loop here indefinitely.

```
9000 GOSUB 10000:IF PEEK(53279)=7 AND STRIG(0)<>0 THEN 9000
9010 GOTO 200
```

The initial routine used at the beginning of each game to play each of the eight notes of the octave while cycling the checkerboard cursor through each of the eight rectangle positions.

```
10000 Z=Z+1:IF Z>8 THEN Z=1
10010 SOUND 0,M(Z),10,8:POSITION 8,5:?" #6;"TUNE";:POSITION 9,6:?" #6;"IN";:POKE 712,16#Z
10020 POSITION J(Z),K(Z):?" #6;"YY";:PO
```

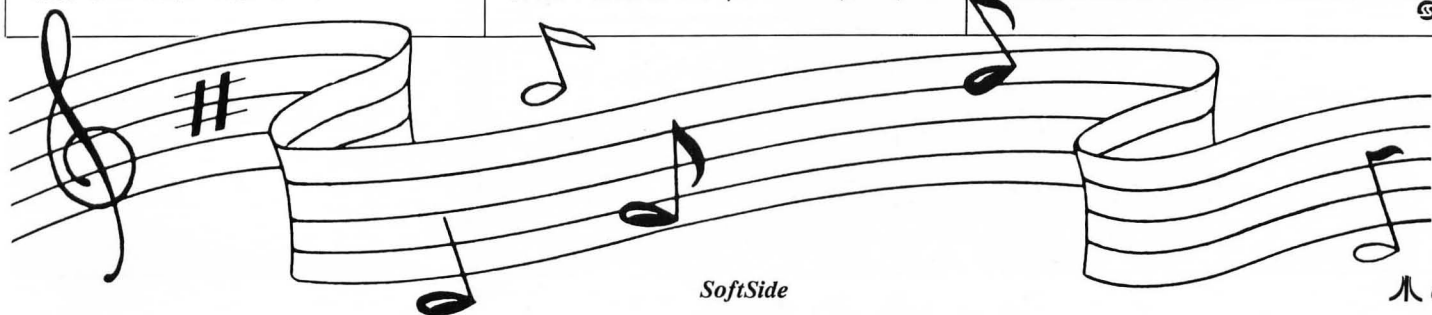
```
SITION J(Z),K(Z)+1:?" #6;"YY";
10030 FOR Y=1 TO 25:NEXT Y:POSITION J(Z),K(Z):?" #6;" ";;POSITION J(Z),K(Z)+1:?" #6;" ";;GOSUB 110
10040 SOUND 0,0,0,0:FOR Y=1 TO 10:NEXT Y:RETURN
```

The Machine Language routines to move the character set into RAM and clear out the player/missile area of memory is initialized. As it is only accessed once, it has been placed at the end of the program.

```
20000 FOR Z=1 TO 32:READ ZZ:POKE ADR(X$)+(Z-1),ZZ:NEXT Z:FOR Z=1 TO 24:READ ZZ:POKE ADR(Y$)+(Z-1),ZZ:NEXT Z
20010 DATA 104,104,133,213,104,133,212,104,133,215,104,133,214,162,4,160,0,177,212,145,214,200
20020 DATA 208,249,230,213,230,215,202,208,240,96
20030 DATA 104,104,133,213,104,133,212,162,8,169,0,160,0,145,212,200,208,251,230,213,202,208,244,96
20040 RETURN
```

ATARI® SWAT TABLE FOR: TUNEIN

LINES	SWAT CODE	LENGTH
3 - 12	AV	622
13 - 60	LU	587
70 - 130	BG	508
140 - 300	YH	518
310 - 430	NR	523
440 - 1055	MB	505
1060 - 1140	JQ	557
1150 - 1270	KN	518
1280 - 1420	BC	585
1500 - 9000	DQ	559
9010 - 20000	VB	517
20010 - 20040	JN	233



K-Byter

Pictures at an XIO-bition

An ATARI® K-Byter by David Suwala, Flanders, NJ.

The classic framework for these pictures is made by plotting squares whose sides are the sum of the next two, smaller squares. This frame, called the golden rectangle, is the basis for many works of art.

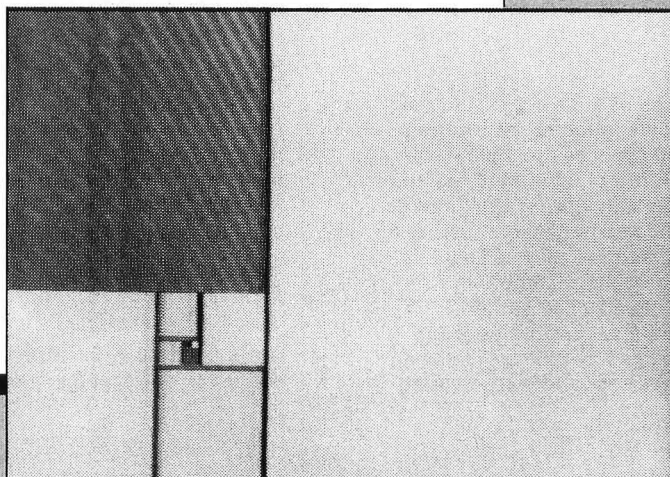
The special fill application command (line 150) is ideal for coloring the square areas in the picture. The SETCOLOR statements select a palette of colors which are randomly painted (line 140) on the screen. If the distortion of the squares offends your sense of artistic proportions, you can correct for the overscan on your TV by multiplying the Y coordinates in line 130 by 0.9 or so.

After viewing your first Mondrian-like masterpiece, press START for an entirely different color combination.

```
10 REM PICTURES AT AN XIO-BITION
20 REM BY DAVID SUWALA
30 GRAPHICS 7+16
40 X0=50:Y0=65:A=1:B=1:F=1:RATCHET=0
50 COLOR 1
60 SETCOLOR 0,3,5:SETCOLOR 1,7,9:SETCO
LOR 2,15,13
```

```
70 RATCHET=RATCHET+1:DN RATCHET GOTO 8
0,90,100,110
80 X0=X0+A:Y0=Y0-B:GOTO 120
90 X0=X0+F:Y0=Y0+A:GOTO 120
100 Y0=Y0+F:GOTO 120
110 X0=X0-B
120 IF RATCHET=4 THEN RATCHET=0
130 PLOT X0,Y0:DRAWTO X0,Y0-F:DRAWTO X
0-F,Y0-F:POSITION X0-F,Y0:POKE 53279,0
140 POKE 765,INT((3#RND(0))+1)
150 XIO 18,#6,0,0,"S:"
160 A=B:B=F:F=A+B
170 IF F=144 THEN 190
180 GOTO 70
190 START=53279:IF PEEK(START)=7 THEN
190
200 GOTO 30
```

Ⓢ



GET SERIOUS . . .

Uncompromised design delivers superior quality and reliability. Today's latest technology allows your Atari 400 to run up to 50% cooler and provide truer video clarity. We guarantee it.

So let's get down to business.

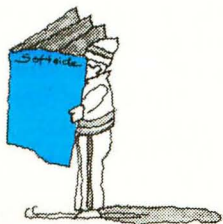
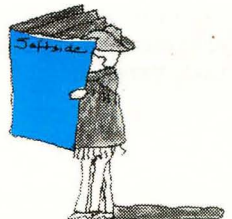
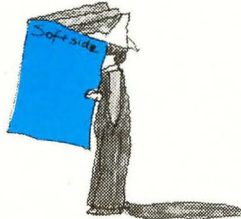
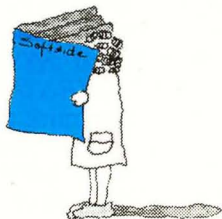
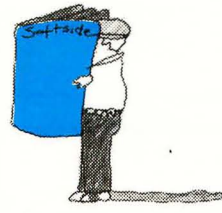
48K RAM for the ATARI 400

 **Tara**
Computer Products INC.

Send certified check or money order. Visa and Mastercard welcome. N.Y. residents please include sales tax. Dealer inquiries invited. Atari is a registered trademark

In USA - 3648 Southwestern Blvd., Dept. S Orchard Park, N.Y. 14127 Tel: (716) 662-7219

In CANADA - 2 Robert Speck Parkway, Suite 1500-S, Mississauga, Ontario L47-1H8 Tel: (416) 273-6820



Subscribe to

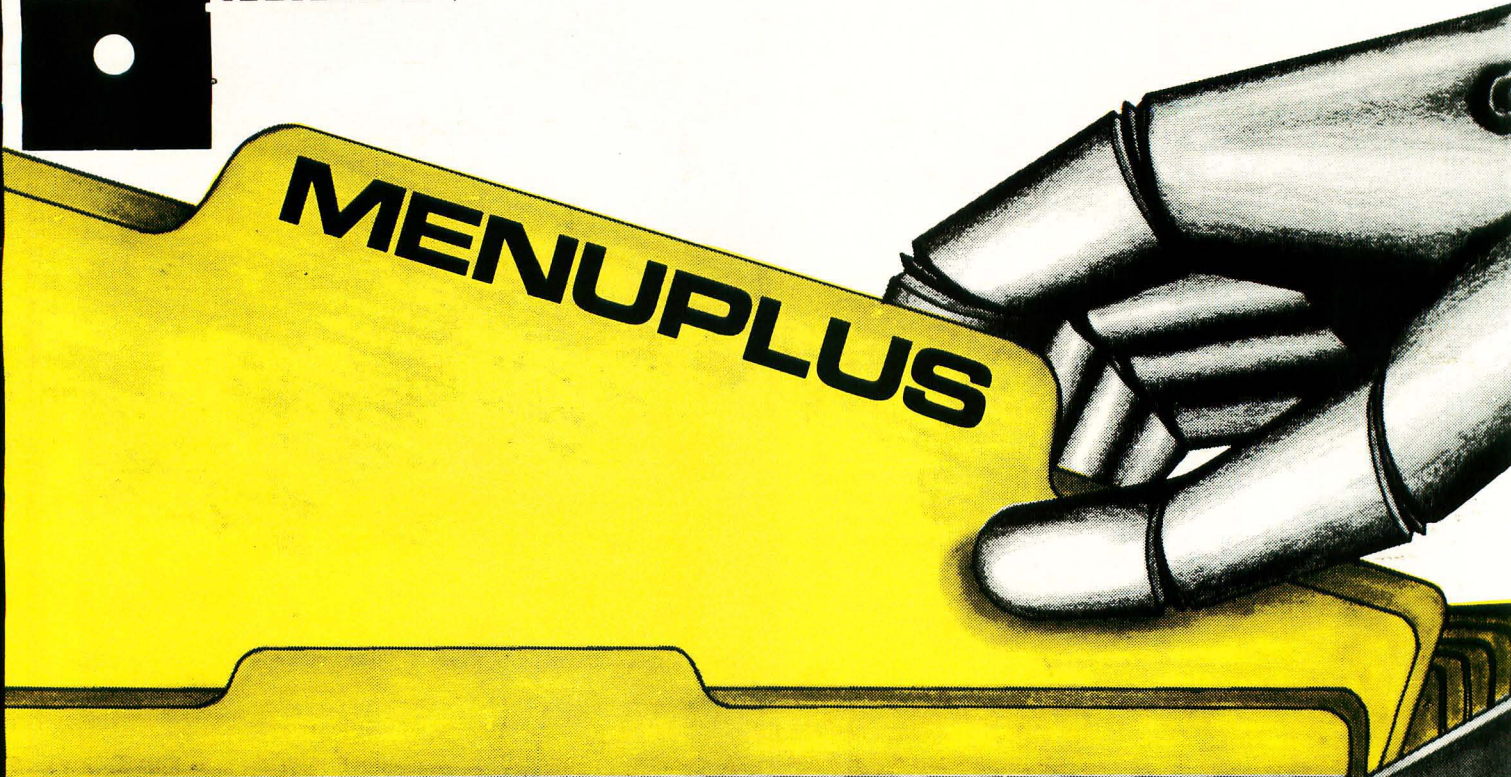
SoftSide

Everyone else has . . .

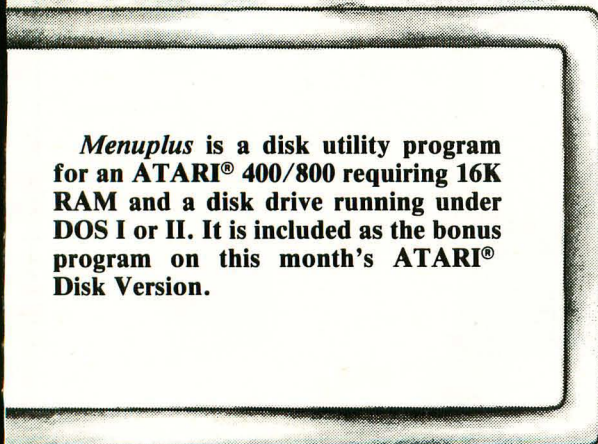
See ordering information on page 16.

 SoftSide

ATARI® DV

A stylized illustration of a hand holding a yellow folder. The folder is labeled 'MENUPLUS' in large, bold, black letters. The hand is rendered in a detailed, shaded style, suggesting a mechanical or robotic nature. The folder is positioned diagonally across the upper half of the page.

MENUPLUS

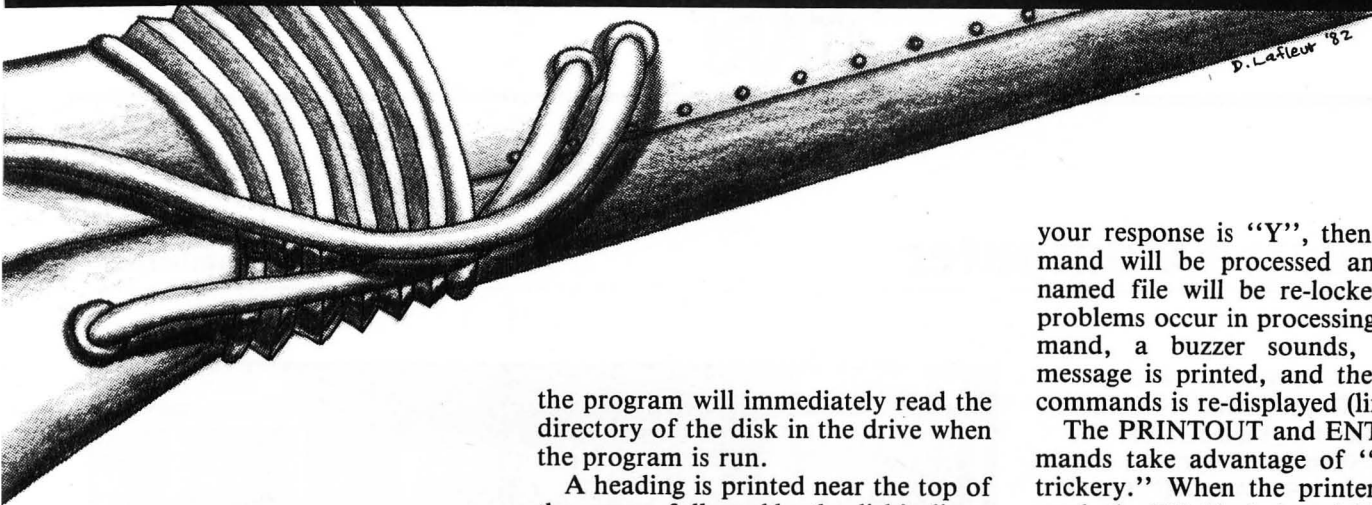
A computer monitor is shown in the lower-left quadrant of the page. The screen displays a block of text describing the 'Menuplus' program. The monitor is a simple, rectangular design with a dark frame.

Menuplus is a disk utility program for an ATARI® 400/800 requiring 16K RAM and a disk drive running under DOS I or II. It is included as the bonus program on this month's ATARI® Disk Version.

by Paul Marentette

Those ATARI® owners who eagerly awaited the release of DOS II were rewarded for their patience. DOS II, unlike DOS I, supports the NOTE and POINT commands for random access of disk files. In addition, the newer DOS uses less memory, as only half of DOS resides in RAM at all times. The latter feature makes more RAM available for BASIC programs, but, at the price of losing immediate access to the DOS menu. When you call up the DOS menu there is an annoying delay while the DOS utilities package (DUP.SYS) is loaded from the disk.

MENUPLUS was designed to minimize the need for the DOS menu. It gives you most of the DOS utilities from BASIC and makes their execution much easier. Wouldn't it be a pleasure to have your disks automatically boot a program which would: 1) load faster than the DOS menu; 2) identify the title and volume number of the disk; 3) neatly format the disk's directory on the screen; 4) number all file names in the directory; and 5) permit you to LOAD, RUN, LOCK, UNLOCK, DELETE, RENAME and ENTER files simply by giving a one-letter command?



MENUPLUS has all of these capabilities and, if you have a printer, will even make a printed listing of the disk's directory.

Most of this is accomplished by using ATARI® BASIC's XIO command. XIO is a generalized input/output command that, among other things, allows DOS type commands to be executed from a BASIC program. Commands are specified by using a numeric code for the desired action. For example, to delete a file you would use the number 33, coding it as follows:

```
XIO 33,#1,0,0,"D:filename"
```

The "#1" is a channel, or IOCB number like those used in OPEN, CLOSE, PRINT, INPUT, GET and PUT commands. The two zeros hold places for unused parameters. Page 30 of the *ATARI® Basic Reference Manual* contains a complete listing of the XIO command numbers.

MENUPLUS takes all the drudgery out of using XIO commands and eliminates the need to type in complete file names when using them. Furthermore, the disk's directory is always present on the screen while you are choosing a command and file number. Part of the directory does not get overwritten with *MENUPLUS*, as it does when using the DOS II menu. Also, unlike DOS II, the directory is automatically re-displayed after each command is executed.

A Closer Look

When first run, *MENUPLUS* waits for you to place a disk in the drive (in case you want to change disks before proceeding). The pressing of any key (RETURN is suggested on the screen) prompts the program to continue. You can omit this step by putting a REM at the beginning of line 60, in which case

the program will immediately read the directory of the disk in the drive when the program is run.

A heading is printed near the top of the screen, followed by the disk's directory in a neatly formatted, two-across fashion. You can customize the heading by altering the second string in line 40. I have *MENUPLUS* stored on each of my disks and have put a disk title and volume number in this location.

The directory is accessed by opening an IOCB in the #6 read directory mode (line 70) and the file names are stored in the simulated string array FN\$. A number is assigned to each file name and is printed to the left of the name. The sector count for each file is printed on the right.

File numbers are used to refer to files throughout the program (great for slow or inaccurate typists!). The PEEK(84) in line 110 checks to see if the cursor has reached the 19th line, and if it has, the program asks for permission to continue the directory on a new screen page.

Once the directory has been listed, the menu of available utilities is displayed. By saving the position of the menu's first line in the ROW variable (line 160), when necessary, the program can clear the screen from ROW down — the directory is never removed from the upper portion of the screen.

Commands are entered using one-letter codes. Invalid commands are trapped (line 410) and then the menu is re-displayed. After a valid command is entered, you will be prompted for a file number (unless you choose the "P" — Printout command). The program searches for the corresponding file name and then carries out the requested action.

If you are deleting a file, you will be asked to approve the action before it takes place (line 300). If you are renaming a file you will be prompted to provide a new file name (line 340). If a file to be deleted or renamed is locked, you are asked if you want to delete/rename it anyway (line 370). If

your response is "Y", then the command will be processed and the re-named file will be re-locked. If any problems occur in processing the command, a buzzer sounds, an error message is printed, and the menu of commands is re-displayed (line 420).

The PRINTOUT and ENTER commands take advantage of "ATARI® trickery." When the printer is to be used, the DV (device) variable is set to 1 and IOCB #1 is OPENED for output to the printer. Then, when the directory print routine is used (lines 70-140), all the "? #DV" commands send output to the printer. Otherwise DV is 0 and IOCB #0 is always open for output to the screen editor. ENTER is used to load a file that has been saved to the disk using LIST. ENTERing a file does not clear out any program currently in memory. Hence, *MENUPLUS* first prints the ENTER "D:filename" command on the screen, then clears the memory with NEW and leaves the cursor on the line where the ENTER command is printed. All you do is press RETURN to begin the loading process (line 450)!

Making *MENUPLUS* Autoboot

Here's what I consider to be the best feature of this utility package. If you run the *BOOTMENU* program, it will create an *AUTORUN.SYS* file (*AUTO.SYS* for DOS I) on your disk. This special file will automatically run *MENUPLUS* when you boot the disk. But remember to SAVE the *MENUPLUS* program with the filename of "D:MENUPLUS" because the *AUTORUN* file is looking for that exact program name syntax. The *BOOTMENU* program prompts you to specify 1 or 2 based on whether you are using DOS I or II. Make sure that you have booted DOS I into the system before running *BOOTMENU* if you are writing an *AUTO.SYS* on a DOS I disk. This is necessary, as the binary load file that is created has a different header (the first 2 bytes) for each version of DOS.

MENUPLUS is a most versatile utility program. Combined with the new features of DOS II, it makes the ATARI® a more powerful and friendly computer. 5

Simulated Computer

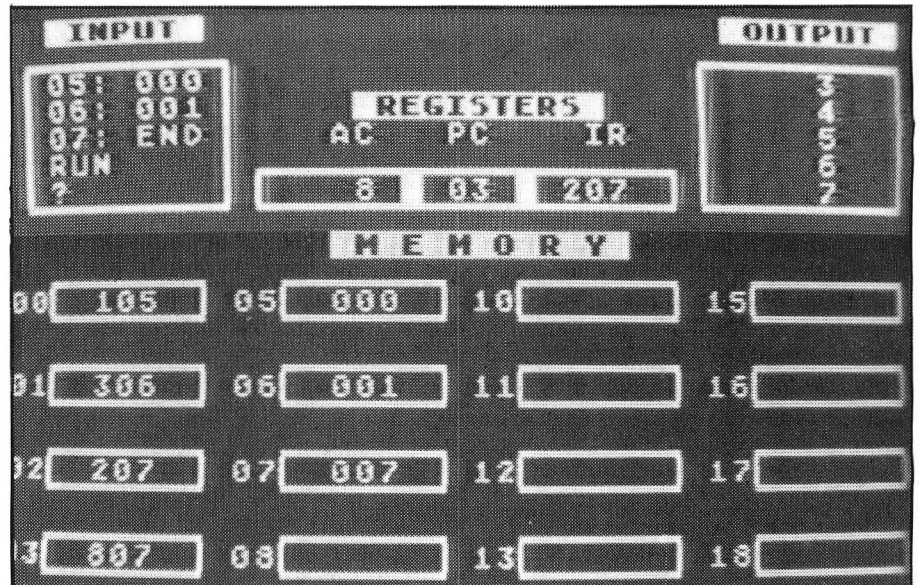
A review by Craig Chamberlain

by James F. Wieder (Edu-Soft, Steketee Educational Software, 4639 Spruce St., Philadelphia, PA 19139) System Requirements: ATARI® 400/800, ATARI® BASIC Cartridge, 16K RAM (Cassette) or 24K RAM (Disk). Suggested retail price: \$14.95 Cassette, \$19.95 Disk.

One application of computers in education is the simulation of a particular event or process. For example, a computer can replay the decisive battles of a war to show tactics, or demonstrate the proper procedure to follow when attempting to operate a nuclear power plant. *Three Mile Island* and *Scram* are good examples. The ATARI® 400/800 computers, with their superior graphics capabilities, have the potential of presenting some very effective simulations. Now, a company has produced a program that makes your ATARI® computer simulate, of all things, a computer! Let's take a look at *Simulated Computer* from Edu-Soft.

The Purpose of a Simulation

The value of a simulator is that it provides a learning experience in a controlled environment. A driving simulator can help teach a person how to drive a car without the risk of an actual crash. *Simulated Computer* applies that idea of safety and convenience to the process of learning about the inner workings of a computer. *Simulated Computer* is a learning aid that presents some essential concepts of how a typical computer operates at the "Machine Language" level. The confusion of using an assembler and the hassle of "crashing the system" when mistakes are made has been eliminated. This program is for the student who wants to learn how to program a computer at the microprocessor level, commonly referred to as "Machine Language." *Simulated Computer* will get the student off to a good start.



A Simplified Computer

A typical microcomputer consists of a microprocessor and memory. The microprocessor, the heart of the computer, generally has many registers and several dozen possible operations. There could be several thousand memory locations. People who want to program in Machine Language often use an assembler and must have an understanding of binary numbers. *Simulated Computer* greatly simplifies this by using only one register (called an accumulator), ten operations, twenty memory locations, and decimal numbers. No assembler is necessary. Error checking is automatic, with explanations of errors. The ten operations include the four primary arithmetic operations (addition, subtraction, multiplication and division), loading and storing the accumulator, performing input and output, jumping to a new instruction, and skipping, on condition, past one instruction. Even with this limited vocabulary, it is possible to do a lot of computing.

Simulated Computer in Action

The main program display consists of a window for entering com-

mands and programs, a program counter, instruction register, accumulator, twenty memory locations, and a comment field. The first thing the student must do is enter a program into memory. Several example programs are provided. Once the program is loaded, execution is started with the RUN command. *Simulated Computer* then shows, step-by-step, the program counter being incremented, an instruction being fetched from memory, and the instruction being executed. At each step, the comment field contains a description of the current activity of the computer. Each register or memory location is highlighted when it is in use. Execution speed can be changed by command or by using the joystick. Better yet, it is possible to single step through a program by pressing a key to execute each step. The program can be stopped at any point and then continued. Any time the student has a question, typing "HELP" will display a number of screens that list all the operations and error codes. The student has full control over the program execution and is kept informed of the computer's activities at all times.

ATARI®

The Tutorial

Simulated Computer comes with a user's guide that is actually a tutorial. The manual I examined presented five lessons with programs, and appendices briefly explaining the terms, user commands, computer instructions and error messages. Although it was nicely produced and free of errors, it assumed the student was familiar with program counters, memory locations, etc.. To avoid those assumptions on future users, the manual is being rewritten with the help of a professional technical writer. When that is done, I expect a concise, complete manual that can be used by anyone who's had limited computer experience.

Field Tested, So To Speak

Simulated Computer has been available on Apple™ and TRS-80® computers for two years, so it certainly has been "field tested." I criticize only a couple of things about the ATARI® implementation. When it boots up, the program displays a "dead screen" a little too long for comfort while it is initializing. Also, switching between the simulation and HELP text messages could be made faster by using two screens in memory simultaneously. But, these are only minor points.

Summary

Simulated Computer from EduSoft has definite educational value. It does what it's supposed to do. Its best application, currently, is in the classroom where there is a teacher present to provide guidance. Hopefully, the revised tutorial will be more suitable to people outside of such a setting. I want to point out that this program does not teach 6502 assembly language programming, used with ATARI® computers. Rather, it presents concepts common to all computers. The program has been submitted to the ATARI® Program Exchange (APX). While this program will not "stun" those who use computers in education, it is a solid addition to the growing collection of educational software for ATARI® computers. ☺



GALACTIC CHASE™

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

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

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

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



Atari 400/800 16k. Written in machine language. Requires joysticks.
Payment: Personal Checks—allow three weeks to clear.

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

Check the dealer in your local galaxy. Dealer inquiries encouraged.
Galactic Chase © 1981 Stedek Software.

SPECTRUM
COMPUTERS

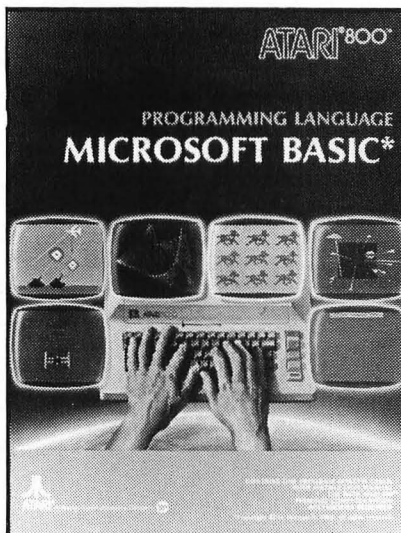
Dept S.
26618 Southfield
Lathrup Village, MI. 48076
(313) 559-5252

ATARI® Microsoft BASIC

A review by Sheldon Leemon

One of the features of the ATARI® Home Computer System on which reviewers often comment is that its BASIC language was not written by Microsoft. This is significant because Microsoft has written the BASIC interpreter for the Apple™, the PET™, the TRS-80®, and so many other popular microcomputers that their version has become the de facto standard of the industry. This lack of a standard BASIC may have given some potential purchasers pause when considering the ATARI®. Now, however, ATARI® has changed this state of affairs by adding *Microsoft BASIC* to the growing list of languages available for use with their Home Computer System.

Just what this will mean to ATARI® owners remains to be seen. One result will be a somewhat greater software compatibility between the ATARI® and other popular personal computers. The instruction manual that comes with *ATARI® Microsoft BASIC* (I'll refer to it hereafter as *AMB*) even has appendices with instruction for converting programs from the PET™, Apple™, and TRS-80® to the ATARI® version. But despite the impression of compatibility that such instructions convey, the fact remains that the ATARI® differs significantly from other microcomputers in its hardware features, most notably in its graphics and sound capabilities. Programs which were written for other computers will not be able to take advantage of the increased user-friendliness that a generous helping of ATARI® sound and graphics can lend to a program, without substantially reworking the code. While the implementation of *Microsoft BASIC* on the ATARI® will probably make available many useful and instructive programs, a thorough knowledge of the workings of the ATARI® computer will still be necessary to adapt these programs



“ATARI® is already working on the next revision, which will be packaged in one, 16K ROM cartridge, with a boot-up file of add-ons for disk users. This will allow owners of minimum configuration systems to use *AMB* with a minimum of fuss.”

to take full advantage of the ATARI®'s unique qualities.

Of course, the potential buyer will want to know not only what new commercial software *AMB* will make available for the computer, but also what advantages this BASIC dialect offers to the programmer who wants to write custom applications. This review will try to highlight some of the more important features of *AMB* in order to help you decide whether it suits your own personal needs.

The initial requirement is that you have a disk drive and at least 40K RAM. If not, *AMB* is definitely not for you — yet. The only version currently available is disk-based, and takes up about 19.5K of memory. With DOS booted up, that leaves

only about 21K free in a 48K system (less if the RS-232 handler in the 850 interface is also booted up). Also, the disk is copy-protected, so you will always have to boot up your master copy and will not be able to make back-up copies. You will also probably not be able to use this version with any of the new disk drives that require a patched version of DOS, including the *Axlon RAM-DISK* board. Another side-effect is that in order to go to DOS, a MEM.SAV file must always be written, further slowing down the process of changing environments. If you find these limitations discouraging, don't give up just yet. ATARI® is already working on the next revision, which will be packaged in one, 16K ROM cartridge, with a boot-up file of add-ons for disk users. This will allow owners of minimum configuration systems to use *AMB* with a minimum of fuss.

Before discussing the specific commands offered by *AMB*, a few words about Microsoft BASICs, in general, are in order. A number of features in *Microsoft BASIC* are implemented quite differently than in *ATARI® BASIC*. Those who have some experience with *Microsoft BASIC* on other machines will be quite comfortable with this version. Because it is based on the full, extended *Microsoft BASIC* model, it most closely resembles *TRS-80® Level II*, with a few more features. Those who have programmed primarily with *ATARI® 8K BASIC* are likely to notice a number of Microsoft's characteristic quirks as soon as they start to enter a program from the keyboard. In *Microsoft BASIC*, no abbreviations are allowed, except the “?” for PRINT. No syntax checking is performed at the time of line entry, so errors in entry will not be apparent until the program is running. Also, *Microsoft BASIC* is much more particular about spacing than the 8K version, and failing to

separate statements from numbers will result in a syntax error. When you do get the error message, however, you will be pleased to find that it is in plain English. Many statements can only be executed in the deferred mode, not from the keyboard.

Microsoft BASIC lets you choose the precision of numeric variables. Integer variables (indicated by the % as the last character in the name) can be chosen for speed, or double precision (indicated by a # at the end) can be used to allow greater accuracy in mathematical calculations. For convenience, the DEF—commands (e.g. DEFINT and DEFDBL) allow you to designate all variables starting with one letter as having the same precision. Math functions are implemented by the interpreter, not by the slow, floating-point package in the OS ROM, which makes faster calculations possible. Also, logical operators such as AND and OR do true bitwise comparisons, as opposed to the *ATARI® BASIC* operators, which compare the truth of the whole expressions. Logical true equals -1, not 1.

The system commands that *Microsoft BASIC* offers for “housekeeping” are very convenient. AUTO allows automatic line numbering for program entry and will warn if a new line number duplicates an existing line number. DEL allows you to delete whole blocks of numbered statements. LIST allows open-ended listing such as “500-” to list statements with a higher line number than 500. LOAD will load programs saved in either SAVE or LIST format. RENUM offers complete renumbering of programs, including references in such statements as GOTO, GOSUB, etc.. Commands such as LOCK, UNLOCK, NAME and KILL allow you to perform the DOS functions that require an XIO command in *ATARI® BASIC*. Additionally, TRON and TROFF allow the tracking of program execution by printing, on the screen, the line number of the statement currently being executed.

There are a number of significant additions to the general program statements offered by *AMB*. For ex-

ample, the MOVE statement allows you to copy any number of bytes from one location in memory to another. This is helpful in animating Player-Missile graphics, and in moving the ROM character set into RAM, so that user-created fonts may be employed. The IF...THEN sequence now allows an ELSE clause which will execute when the IF clause is not true. OPTION BASE allows you to choose whether array subscripts will start with a 0 or a 1. The WAIT command allows you to pause the program until a location in memory takes on a specific value. This is ideal for halting execution until VBLANK occurs, so that graphics changes can

“A rather remarkable addition in *AMB* is the COMMON statement. This allows you to designate certain variables to keep their values from one program run to the next. This greatly facilitates chaining several related programs together to operate from one menu program, and will be most helpful in overcoming the RAM limitations imposed by the size of the interpreter.”

be made without disrupting the display. It could also be used in conjunction with the real-time clock for a measured pause. There is another time-related statement which is even more fascinating. This is the AFTER command, which allows you to change the flow of program control after a given period of time. For example, the statement AFTER (600) GOTO 200 would have no immediate effect on the program. But in 10 seconds (600 “jiffies,” or 60ths of a second), the program would stop doing whatever it had been doing and start executing at line 200! This gives you, in effect, a time-driven interrupt.

A rather remarkable addition in *AMB* is the COMMON statement.

This allows you to designate certain variables to keep their values from one program run to the next. This greatly facilitates chaining several related programs together to operate from one menu program, and will be most helpful in overcoming the RAM limitations imposed by the size of the interpreter. Other new program statements include ON ERROR, a slightly different error trapping scheme than TRAP, which allows you to RESUME the program after an error at the line where the error occurred, the next line, or at any designated line number. ERL will return the line number where an error occurred, while ERR will let you generate any error, for purposes of debugging your program’s error-trapping code. OPTION PLM, OPTION CHR, and OPTION RESERVE statements let you set aside reserved areas of memory for player-missile graphics, character sets, or machine code.

I/O operations have been substantially overhauled in this *BASIC*, particularly as they relate to *ATARI®*’s unique system for handling I/O in a device-independent manner through the use of a Central I/O Utility. For example, the OPEN command used to assign an I/O channel to a device now has a syntax closer to English (e.g. OPEN #1, “K:” INPUT). However, there is a price to pay for this simplification. Because the command syntax no longer corresponds directly to the requirements of the Central I/O Utility, you can only OPEN a device for the READ, WRITE, UPDATE and APPEND functions, and not to read the Directory. You cannot OPEN the cassette for reading or writing files with short inter-record gaps. This would not be so bad if the XIO command had been retained. But, because most of its functions have been implemented through other commands, it has been deleted. The only concession made to the *BASIC* user wishing to perform missing CIO functions such as formatting a disk, reading the disk directory, or reading or writing a block of memory to or from a disk file, is the inclusion of a disk file called CIOUSR. This file provides three prewritten USR routines

ATARI® OWNERS & PROSPECTIVE OWNERS

Now is the time to buy
ATARI® products at low,
low prices!

ATARI® 800 w/ 16K	\$699
w/ 32K	\$784
w/ 48K	\$869
ATARI® 400 w/ 16K	\$350
ATARI® 850 Interface	\$175
ATARI® 830 Modem	\$159
ATARI 810 Disk Drive	\$460
ATARI 410 Recorder	\$79

Cartridges \$35 each, any three for \$99.95

Asteroids • Centipede • Missile Command • Pac-Man • Space Invaders • Super Breakout • Computer Chess

PILOT (COMPLETE)	\$99
PILOT HOME PKG	\$63
STAR RAIDERS CART	\$38
CAVERNS OF MARS	\$31
MUSIC COMPOSER CART	\$45
ENTERTAINER PKG	\$83
PROGRAMMER PKG	\$56
COMMUNICATOR PKG	\$320

PKG A — ATARI® 800 w/48K, 850 Interface, 810 Disk Drive, Epson MX-80 Printer w/ GRAFTRAX & Cable	\$1990
PKG B — ATARI® 800, w/16K, 410 Program Recorder, Any 3, \$35 Cartridges & Pair of Joysticks	\$877
PKG C — ATARI® 400 w/16K, 410 Program Recorder, Programmer Pkg	\$475
PKG D — ATARI® 400 w/16K, Entertainer Pkg, Any 2, \$35 Cartridges	\$494

Add 1% for shipping & handling
Michigan residents add 4% sales tax.
ATARI® & are trademarks of ATARI® Inc
WE ALSO BUY AND SELL USED ATARI®
EQUIPMENT. WRITE TO US.

Computer's Voice™ software for the ATARI® 400/800 computers

Word Search Puzzle Maker — Makes word search puzzles and answers on a printer-24K cassette, 24K disk \$24.95.

MENUMAKR — takes in and dumps text screens in GRAPHICS 0-8K cassette, 16K disk \$24.95.

MATHFAKS — Math drill program using color and sound. Optional accessories include a printer and VOTRAX Type-N-Talk-24K cassette, 24K disk \$24.95.

Electronic Grade Book — Great classroom aid for teachers. Uses number or letter grades, weighted scores, screen or printer output, and more. 32K cassette, 40K disk \$64.95.

Light Pen — for the ATARI® 400/800 by Symtec. Suggested list price \$149.95. **Mention this ad and pay \$134.95.**

AXALON — 128K RAMDISK for ATARI® 800s only \$420.

Add \$2 p/order for shipping
Michigan residents add 4% sales tax

Computer's Voice™
2370 Ella Dr.
Flint, MI 48504
(313) 238-5585

ATARI®

which will allow you to condition the I/O Control Blocks and then call the Central I/O Utility. However, the user who is sophisticated enough to take advantage of the CIOUSR file is probably sophisticated enough to POKE the right values directly into the IOCB, and control all CIO with his or her own USR statement. The upshot is that some I/O flexibility has been sacrificed for the sake of easy use.

In other areas, I/O flexibility has been greatly enhanced. The INPUT command will let you substitute your own prompt message for the default question mark, and LINE INPUT will let you input a complete line, ignoring commas, quotation marks, and other terminators. There are a number of handy statements to aid in the production of neatly formatted output. SPC will print a designated number of spaces. TAB will let you tab to a given print column. PRINT USING is fully implemented, letting you right justify, insert decimal places, trailing spaces, leading spaces, commas, dollar signs, asterisks, and generally line everything up in nice, neat, uniform columns. For screen output, the POSITION command has been replaced with PRINT AT(X,Y), where the variables X and Y indicate the screen row and column. Interestingly enough, this command also replaces the statement, POINT, when used with a disk file as the device — in that case, the variables X and Y indicate which sector and byte should be written.

The function library in *AMB* is mostly the same as *ATARI® BASIC*, although the math routines will generally have greater accuracy and better speed. Trig functions are only available in radians in *AMB*. TAN is implemented. RND is a little more flexible, allowing you to generate random integers and repeatable pseudorandom sequences as well. TIME is added as a function which returns elapsed time to 1/60 of a second. The USR command is somewhat different in *AMB*. Rather than passing arguments to the machine language routine on the stack, *AMB* passes only one integer argument directly to two zero-page locations. While a programmer can

use RAM from \$CD to \$FF in a USR call, there are no "safe" user zero-page locations, where variables can be stored immune to meddling from BASIC. As a matter of fact, not even page 6 is sacred — BASIC uses half of it, leaving little protected space for non-relocatable code.

One of *AMB's* great advantages over *ATARI® BASIC* is its ability to accept user-defined functions. This lets the user, in effect, make up his or her own BASIC commands. For example, as mentioned below, there is no *AMB* statement comparable to STICK(X), which returns the value of the joystick. However, the user could define such a function with the line DEF STICK(X)=PEEK(632+X). Then, every time thereafter in a program STICK(X) was used, the function would return the value of joystick number X. Moreover, the user could even define HSTICK(X) and VSTICK(X) as in BASIC A+, where the functions would return a 1, 0, or -1 depending on whether the joystick is being pushed up or down, right or left. One of the more serious oversights in the manual is the lack of attention to this command. The functions it allows you to define not only save time over a subroutine call, but can make possible the use of program constructs which otherwise could not be used. The DEF statement also allows you to define string functions, which will perform any allowable combination of string manipulations.

Perhaps the most significant difference between *AMB* and *ATARI® BASIC* is in the area of string handling. In *ATARI® BASIC*, strings are one-dimensional. They must be DIMensioned, and can be as long as memory allows. In *AMB*, one-dimensional strings don't have to be DIMensioned, and true string arrays are implemented. The latter feature is greatly prized by *Microsoft BASIC* enthusiasts, who find that string arrays much simplify the task of character data manipulation. Maximum string length is limited to 255 characters, however.

If the amount of free memory gets low, *AMB* performs what is

continued on page 72

ATARI®

COMPARISON OF BASIC COMMANDS

Name	Microsoft	ATARI®	BASIC A +	Name	Microsoft	ATARI®	BASIC A +
System Commands							
AUTO	X		X	OPTION CHR	X		
BYE		X	X	OPTION PLM	X		PMGRAPHICS
CLEAR	X	CLR	CLR	OPTION RESERVE	X		
CLEAR STACK	X			PADDLE, PTRIG		X	X
CLOAD	X	X	X	PEEK	X	X	X
CONT	X	X	X	PLOT			
CSAVE	X	X	X	PLOT (X,Y) TO (X,Y)	X	DRAWTO	DRAWTO
DEL	X		X	PMMOVE, PMWIDTH			X
DIR			X	POKE	X	X	X
DOS	X	X	X	POP		X	X
KILL	X		ERASE	PRINT	X	X	X
LIST	-nn,nn-	X	X	PRINT AT	X	POSITION	POSITION
LOAD	X	X	X	PRINT USING	X		X
LOCK	X		PROTECT	PUT	X	X	X
LOMEM			X	RGET, RPUT			X
LVAR			X	READ	X	X	X
MERGE	X	ENTER	ENTER	REM	X	X	X
NAME	X		RENAME	RESTORE	X	X	X
NEW	X	X	X	RESUME	X		
RENUM	X		X	SOUND	X	NOT TIMED	NOT TIMED
RUN	X	X	X	SPC	X		
SAVE	X	X	X	STACK	X		X
SAVE w/LOCK	X			STATUS	X	X	X
TROFF	X		X	STICK, STRIG		X	X
TRON	X		X	STOP	X	X	X
UNLOCK	X		UNPROTECT	TAB	X		X
VERIFY	X			WAIT	X		
BASIC statements				XIO		X	X
AFTER	X			WHILE, ENDWHILE			X
BPUT, BGET			X	String Functions			
BUMP			X	ASC	X	X	X
CLOSE	X	X	X	CHR\$	X	X	X
CLS	X			INKEY\$	X		
COLOR	X	X	X	INSTR	X		X
COMMON	X			LEFT\$	X		
DATA	X	X	X	LEN	X	X	X
DEF				MID\$	X		
DEFINT	X			RIGHT\$	X		
DEFDBL	X			SCRN\$	X	LOCATE	LOCATE
DEFNG	X			STR\$	X	X	X
DEFSTR	X			STRING\$	X		
DPEEK, DPOKE			X	TIMES	X		
DIM	X		X	VAL	X	X	X
END	X	X	X	VARPTR	X	ADR	ADR, PMADR
EOF	X			BASIC Functions			
ERROR	X			ABS	X	X	X
FILL	X	XIO 18	XIO 18	ATN	X	X	X
FOR..TO..STEP...,NEXT	X	X	X	CLOG		X	X
GET	X	X	X	COS	X	X	X
GOSUB..RETURN	X	X	X	DEG		X	X
GOTO	X	X	X	ERL			ERR(1)
GRAPHICS	X	X	X	ERR	X		ERR(0)
HSTICK, VSTICK			X	EXP	X	X	X
IF..THEN..ELSE	X	NO ELSE	UNLIMITED ELSE	FRE(X)	X	X	X
INPUT	X	X	X	INT			
LET	X	X	X	LOG	X	X	X
LINE INPUT	X			RAD		X	X
LPRINT		X	X	RANDOMIZE	X		
MISSILE			X	RND(X)	X	0<X<1	0<X<1
MOVE	X		X	SGN	X	X	X
NOTE				SIN	X	X	X
ON ERROR	X	TRAP	TRAP	SQR	X	X	X
ON..GOSUB	X	X	X	TAN	X		
ON..GOTO	X	X	X	TIME	X		
OPEN	X	X	X	AND, OR, NOT, XOR	X	NO XOR	NO XOR
OPTION BASE	X		SET				

known as string-gathering or garbage collection, by which it compresses the strings down to the space actually occupied by data. The programmer should be aware of this feature, as it may cause a noticeable pause in a program while it occurs. As an offshoot of this process, strings tend to move around in memory while a program is running, making them a less dependable place in which to hide machine-code subroutines, player-missile graphics, etc..

In addition to string arrays, many other handy string features are implemented. There is a true concatenation operator ($C\$ = A\$ + B\$$). `LEFT$`, `MID$`, and `RIGHT$` help separate substrings, and `INSTR` performs a search for a substring within a larger string. `STRING$` allows you to fill a string with any number of repetitions of a single character, or string of characters. Other functions now included within string operations are `INKEY$`, which records a keypress on the fly, `TIME$` which returns the time in Hours : Minutes : Seconds format, and `SCRN$`, which replaces `LOCATE`, returning the value of data under the graphics cursor.

In the area of graphics, `AMB` is very similar to `ATARI® BASIC`. Major changes include the combination of `PLOT` and `DRAWTO` into a `PLOT (X,Y) TO (X,Y)` command that can be chained indefinitely within a single statement — a much streamlined procedure over the `ATARI® BASIC` version. A `CLS` command has been included to clear the screen. Also, `SETCOLOR` has been expanded to include the registers for player-missile graphics.

Player-missile graphics are supported to a moderate extent by `AMB`. The `OPTION (PLM)` commands set aside space for either single-resolution or double-resolution PM graphics, and put the location of the base address into the proper hardware register. This address can be found with the `VARPTR` command (`AMB`'s answer to the `ADR` function), and can be used to calculate the offsets into player and missile storage areas. As stated above, the `SETCOLOR` command can be used to control player color. The `MOVE`

command can be used to shift the offset position of a player, to achieve vertical movement. But player width, priority control, horizontal position, collision detection, and other features still must be implemented through `PEEKs` and `POKEs`. Here is another area where the `DEF` function could give the user a lot of aid. The manual thoroughly documents the features of PM graphics, and gives plenty of step-by-step examples for the beginner. This is a big improvement over the `ATARI® BASIC` manual, which does not acknowledge the existence of PM graphics. However, this manual still ignores the existence of the GTIA chips, and graphics modes 9, 10, and 11 which it makes available to users of machines made after 1981.

Last, but not least, we come to game controllers and sound. As mentioned above, there are no commands for joystick, paddle, or lightpen reading, although they can easily be implemented with `DEF` functions or `PEEKs`. There has been, however, a pleasant addition to the `SOUND` statement — a fifth parameter that designates the duration of the note in 60ths of a second. This means that you no longer have to write imprecise delay loops to sustain sound effects.

The manual supplied with `AMB` is worthy of some comment. It is somewhat sketchy on the features of `AMB` that are common to all Microsoft implementations, so that some prior familiarity with those features would really help the reader. Such information is readily available from any of the great many books devoted to `BASICs` such as the one used by the `TRS-80®`. On the other hand, as far as `ATARI®` specific features go, the manual is most detailed, giving explanations and examples concerning player-missile graphics, alternate character sets, a complete memory map, and conversion information for translating programs from other `BASICs`.

So there you have it. `ATARI® Microsoft BASIC` is large and feature-laden. Though not quite bug-free yet, there is every reason to hope that the major bugs will get ironed out before they are frozen

forever into ROM, as are the flaws in `ATARI® BASIC`. Many users will find that its numerous new commands and statements far offset the trade-offs it requires, such as large memory requirements, limited string length, lack of syntax checking at the time of program entry, lack of abbreviations for keywords, etc.. For other users, who would like an extended `BASIC`, but prefer the general format of `ATARI® BASIC`, `BASIC A+` will be a better choice (I personally like `BASIC A+` a lot, and for that reason, I've included it in the comparative table of features offered by the various `BASICs`). But at least now, `ATARI®` owners will not have to stand alone in the crowd of microcomputer owners. They will have access to the full range of programs and techniques offered by `Microsoft BASIC`, if they so choose.

Editor's Comment: At this printing, Mr. Leemon had just received a copy of the ROM cartridge version of `Microsoft BASIC` and has promised an update review in the very near future.

`SoftSide` readers have, in the past, proven themselves to be connoisseurs of the "best of `BASICs`." If `Microsoft BASIC` lives up to current expectations, it will quickly become the standard of `ATARI®` owners. As we do not expect to run another general survey of our readers until sometime next year, it is vital that we find some other means for discovering how many of you have purchased `Microsoft BASIC`. If you currently own a copy of the language, please send us a card so we may have you "on record." If you purchase a copy in the future, please inform us as soon as you have done so. As soon as we feel the number of responses justifies the conversion of our standard to `ATARI® Microsoft BASIC`, we will do so. Be sure to include whether you own the cartridge or disk version of the language and tell us how much RAM you have in your `ATARI®`. Send cards to:

SoftSide Publications
Department AMB
6 South Street
Milford, NH 03055

Exploring the ATARI® Frontier

ANTIC and the Display List: Part II by Alan J. Zett

Note: This is the second part of a series. Each column assumes that the reader has read the previous columns. To avoid confusion, we advise you to start reading the series with the first article.

Looking Back

Last time I presented some of the basics for understanding the Display List. Even though it was sufficient to give a general idea of how the Display List works, a large amount of information still remains to be explained before you can make and use a custom Display List. In fact, so much material remains that I am dedicating this column specifically to the building and using of ONE type of custom Display List. The discussion of mode line options has been postponed until a later time.

The Catch... Revisited!

As I mentioned, the real advantage of ANTIC is its ability to mix text and different types of graphics modes. Modifying the Display List to accommodate a custom display is easy. After planning how the display will look, it merely takes a few POKES from BASIC and the rest is history. What I didn't mention (no need to frighten anyone off too soon!) were the problems that arise when attempting to combine modes requiring different amounts

of display data for each line. Even though a particular mode line may have a resolution of 160 pixels (a pixel being equivalent to one character or graphics block), the actual number of bytes required for that mode line varies with the number of pixels stored in each byte and the number of colors available.

GRAPHICS 0 is the simplest of screen data displays. One byte is required for each character on the display. Comparing GRAPHICS 6 and 7, their resolution is the same, but GRAPHICS 7 requires twice as much memory because it allows twice as many colors. Note also that in most GRAPHICS modes, the number of bytes per mode line (see Figure 1 from last month) is much less than the pixel resolution per mode line (along the X-coordinate). Based on the resolution and the number of bytes required, GRAPHICS 6 packs eight pixels into a byte (one pixel per bit) and GRAPHICS 7 packs in four (one pixel per bit pair). We'll get into the actual format of screen data later in the series.

ANTIC can control a custom display by itself without trouble. However, when used in conjunction with BASIC, some of its versatility is lost as a trade-off for the use of BASIC's screen/video commands. This creates the problem of how to write on each display line effectively without BASIC or Operating System (OS) interference. In some cases, a simple POKE will allow a

POSITION/PRINT or PLOT/DRAWTO command to display information as easily as from a normal Display List. In others, the only way to get the screen data to the place where it is required is to manually POKE it there.

In general, the most time consuming portion of the work required to create a custom Display List is the planning and preparation. Custom displays should be designed while keeping in mind the problems associated with them. In most cases, the display you want to use will be much more complicated than it looks at first.

There are several common problems that will concern you: How many bytes of memory does each mode line require? (Unequal numbers of bytes for each line leads to special problems.) Does the total number of scan lines add up to 192 or less? Will a mode line be put in a position where its Y-coordinate is greater than that which is allowed for a Display List of that type? (For example: A GRAPHICS 0 mode line being placed in the middle of a GRAPHICS 8 display would make the start of the GRAPHICS 0 mode line have a coordinate of, say, POSITION 0,77--which in GRAPHICS 0 would cause a "cursor out of range" error). Can the Display List be optimized to make it easy to use? (This sometimes requires extra effort. After designing the display, there are usually some places that need to be padded out to make the display easy to use.) None of these problems are easy to deal with, so in order to give you an idea of how to create a simple custom Display List, we'll build one.

A Simple Custom Sample

The first rule of thumb when building a Display List is to map out on paper what the screen will look like. The most important things to consider at this stage are the trade-offs required to build a Display List that is 192 scan lines long. For our first try, we'll start with a "title page" effect of mixed text modes.

To boldly announce our title, we'll use the large text mode of GRAPHICS 2 followed by a smaller descriptive phrase in GRAPHICS 1 and a small copyright notice at the

T.H.E. SMART TERMINAL[®]



TURN YOUR ATARI 400 OR 800 INTO
A REAL SMART TERMINAL

Get up to date information from
services like Dow Jones, Compuserve,
The Source, and local timesharing
computers.

Save the information on disk or cassette
for editing or reviewing when you
disconnect from the telephone line!

Send the edited information back to
the timesharing system when you are
ready.

**REDUCE YOUR CONNECT CHARGES
BY READING AND WORKING OFF
LINE!!**

- User Friendly
- Disk or Cassette Based
- Works with Hayes Smart Modem
- X-ON/X-OFF Protocol
- Runs in 16K
- Serial or Parallel Printers
- Menu or Command Driven
- Save Data on Cassette or Disk
- Upload/Download Atari 400 or 800
- Multiple files in memory

This package allows you to define,
transmit and receive characters so you
can send characters and control codes
not found on the Atari keyboard and
receive characters that the Atari can
translate into something it understands.

**A POWERFUL COMMUNICATIONS
PACKAGE AT A SUPER PRICE!**

**T.H.E. MOST Sophisticated Communica-
tions Package Available for the Atari,
400 or 800 and it's available on Cassette,
too!**

\$49.95 cassette or disk

ORDERING INFORMATION

Call BINARY directly to place your
order. Our order lines are open 24
hours per day, 7 days per week.

Shipping and handling charges:

North America: Add \$2.50
Outside N.A.: Add 10%
Michigan Residents: Add 4% tax.

Payment Methods:

VISA, Master Charge, AMEX, cash,
certified check, personal check
(allow for clearance), money order.

**Look for Binary Software Products at
your local computer store.**

Dealer Inquiries invited

BINARY[™]

COMPUTER SOFTWARE
3237 Woodward Ave.
Berkley, MI 48072
(313) 548-0533

BINARY CORPORATION

ATARI[®]

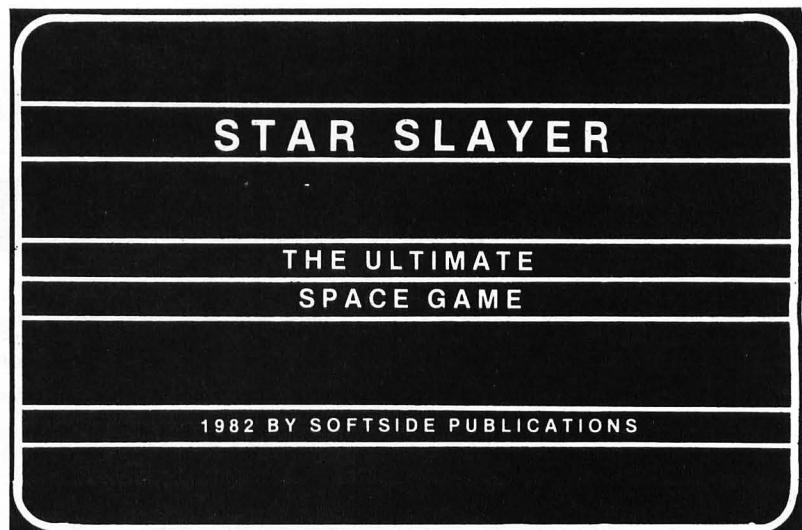


Figure 1

bottom in GRAPHICS 0. Our
scratch pad for figuring the Display
List would look something like
Figure 1.

The first step in "hand assembl-
ing" the Display List is to find out
which of the mode line types being
used requires the most memory.
This is determined by looking up (in
figure 2) the memory requirements
for each BASIC GRAPHICS mode
that corresponds to the mode line
being used in the display. Looking
at GRAPHICS 0 through
GRAPHICS 2 we find that
GRAPHICS 0 requires the most
memory. Therefore, when we build
the display list from BASIC, we will
start by initializing a GRAPHICS 0
Display List and then modifying it.

After the general position of each
mode line is sketched, the next
problem to overcome is the adjust-
ment of the number of scan lines to
192 and the adjustment of the
number of mode lines used in each
portion of the redefined Display
List so that the display is balanced
out. If we were to simply modify
each line we needed to redefine, an
interesting and somewhat irritating
problem would arise.

When we type GRAPHICS 0 at
the start of the program, the
Operating System (OS) generates a
Display List of 24 mode lines con-
sisting of 40 bytes per display line. A
mode line, in this case, is the AN-
TIC code for a screen line. A display
line is the visible physical line which
the OS sees as starting at multiples

of the number of bytes per mode
line.

For example: The first byte of
display line four in GRAPHICS 0 is
located at the beginning of screen
memory plus a number equal to the
sum of the number of bytes per
mode line (40) times the display line
number. This means that in order to
write to the first byte of line four,
the OS must write to the start of
screen memory plus 160 (40*4).
Now if we should change the length
of line one to be GRAPHICS 2 (20
bytes long), the OS would still write
to screen memory plus 160 (it
doesn't know that we have removed
20 bytes from line one). But that
location will be 20 bytes past the real
location of the first byte of line
four. The OS would be looking for
40 bytes of data for mode line one
whereas ANTIC only needed 20.
This would cause the OS to think
that the next 20 bytes following the
20 GRAPHICS 2 bytes are still part
of the same display line.

In effect, half of the next mode
line on the display is actually the
previous display line wrapping
around to the bottom. In order to
write to the second mode line of the
display (still a GRAPHICS 0 line),
we actually have to write to the mid-
dle of the first display line. But this
is only the beginning of the prob-
lem. Every time we try to print at an
X-coordinate of zero following the
modified mode lines, the text ap-
pears to be indented 20 BYTES
(halfway) into the display line.

Figure 3 shows graphically what actually happens. Display line one stretches halfway into mode line two, and every display line after that is offset by 20 bytes.

One way to correct this problem is to add enough extra mode lines of different lengths to bring the start of each line back to the left edge. For example: If we modify two mode lines to be GRAPHICS 2, then all the display lines after the GRAPHICS 2 mode lines will be all right. This is because, by using two lines, the sum of the bytes required for the GRAPHICS 2 lines comes out to be 40. The lines following the GRAPHICS 2 lines are shifted another 20 bytes, and the screen lines up again. The display line numbers will be offset by the number of extra lines required to align the display. Since we have inserted an extra GRAPHICS 2 line into the Display List, all the following display line numbers will be one LESS than they would normally be. The OS thinks that lines one and two are all part of line one. Therefore, display line four will appear to be line three to the OS. ANTIC, however, requires separate mode line bytes for each line on the display, but when we PRINT/PLOT to the screen, it's the OS we have to accommodate, not ANTIC.

As long as redefinition is kept in multiples of the base GRAPHICS mode (i.e., the one used as the base which is modified), then normal screen/video commands will work. Of course nothing is THAT easy. In order to POSITION and PRINT

Figure 2: Memory Requirements for BASIC Graphics Modes.

GR. MODE	TYPE	RESOLUTION	MEM SIZE
0	Text	40 x 24	960
1	Text	20 x 20	560
2	Text	20 x 10	360
3	Graphics	40 x 20	360
4	Graphics	80 x 40	560
5	Graphics	80 x 40	960
6	Graphics	160 x 80	1760
7	Graphics	160 x 80	3360
8	Graphics	320 x 160	6560
17	Text	20 x 24	480
18	Text	20 x 12	240
19	Graphics	40 x 24	240
20	Graphics	80 x 48	480
21	Graphics	80 x 48	960
22	Graphics	160 x 96	1920
23	Graphics	160 x 96	3840
24	Graphics	320 x 192	7680

there are some special things to do, as we'll see later.

Now we should write down the number of bytes per mode line on our scratch pad, and next to that the number of scan lines each mode line takes (see figure 1 last month). GRAPHICS 2 requires 20 bytes per

line at 16 scan lines per mode line. GRAPHICS 1 requires 20 bytes per line at 8 scan lines per mode line. GRAPHICS 0 requires 40 bytes per line at 8 scan lines per mode line. By putting 2 lines of GRAPHICS 2 at the top of the display, and 2 lines of GRAPHICS 1 in the middle, all

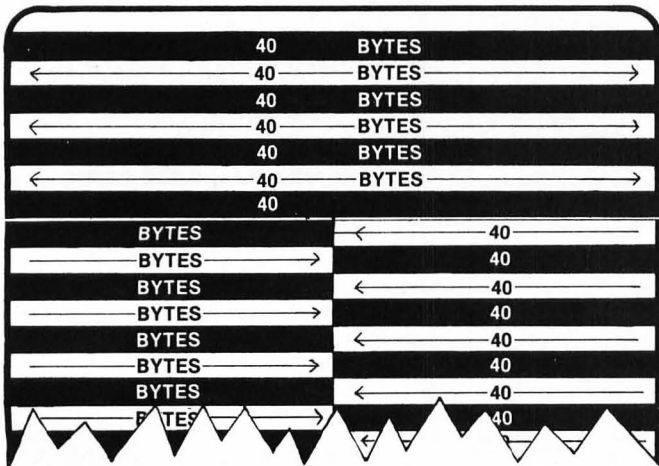


Figure 3

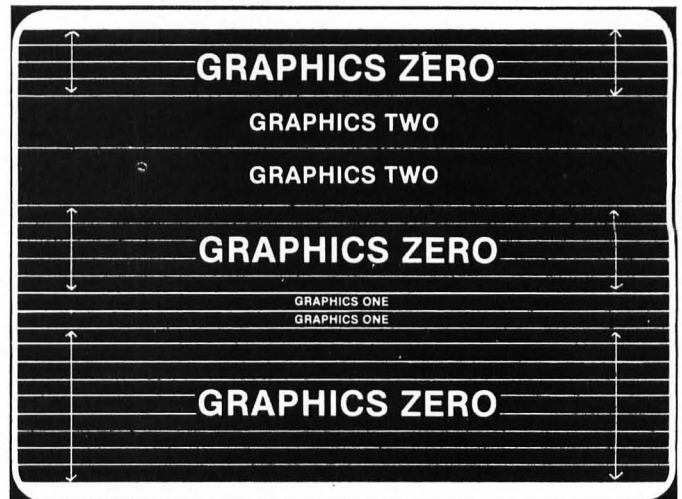


Figure 4

GRAPHICS 0 lines start at the left edge where they belong. The problem now is that when we add up all 24 GRAPHICS 0 mode lines (including the ones we will modify) the scan line count comes out to be 208 — too many. We can solve this by removing excess GRAPHICS 0 lines from the bottom of the Display List. Taking out two GRAPHICS 0 lines will adjust the scan line count to be 192.

Figure 4 is a completed outline of

the custom Display List. All that remains now is to actually modify the standard BASIC GRAPHICS 0 Display List. It would be a good exercise to translate, on paper, what the Display List should look like, and then look at Figure 5. Compare this to the GRAPHICS 0 Display List (Figure 2 last month).

The Modifying Begins

What we want to do now is to

generate a GRAPHICS 0 Display List and modify it to look like Figure 5. We start by executing a GRAPHICS 0 command and PEEKing into memory to find out where the OS has put the Display List. The following lines of BASIC will accomplish this:

```
10 GRAPHICS 0:POKE 752,1
20 DL=PEEK(560)+PEEK(561)*256
```

After these lines have been executed, the variable DL will point to the first byte of the Display List. Since the first three bytes of any Display List are blank mode line instructions (see last month), I usually offset the pointer by four. The variable DL will then point to the location of the first byte of screen data. As an extra benefit, this will make display line number one equal to DL-1 and display lines two and up equal to DL + 2, + 3, and so on. This seems confusing at first, but there is a twisted sort of logic involved if you look hard enough.

Next, we pick the Display List byte that approximately corresponds to the location drawn on our scratch pad. In this case, line five in GRAPHICS 0 is the closest. Then we must calculate the position of the byte to modify, and POKE it with the ANTIC code that corresponds to the GRAPHICS mode we want to use. If you look at Figure 1 from the last installment, you will find that GRAPHICS 2 is known to ANTIC as a mode number 7 byte. By POKEing this into the location we calculated, we will have modified one line on the screen. To defeat the previously discussed conflict that arises from GRAPHICS 2 requiring only 20 bytes of screen data, we can also POKE the next location with 7, and the screen will line up again. We can now add these lines to the program:

```
30 DL=DL+4
40 POKE DL+5,7:POKE DL+6,7
```

Display line 12 is the nearest approximation of the next set of lines to be modified. GRAPHICS 1 is listed as being equivalent to ANTIC

Figure 5: A Modified Graphics Zero Display List

DL BYTE NO.	BYTE VALUE	MODE TYPE
1	112	Blank
2	112	Blank
3	112	Blank
4	66	Graphics 0 w/LMS option
5	nn	Least significant byte of screen memory
6	nn	Most significant byte of screen memory
7	2	Graphics 0
8	2	Graphics 0
9	2	Graphics 0
10	7	Graphics 2
11	7	Graphics 2
12	2	Graphics 0
13	2	Graphics 0
14	2	Graphics 0
15	2	Graphics 0
16	2	Graphics 0
17	6	Graphics 1
18	6	Graphics 1
19	2	Graphics 0
20	2	Graphics 0
21	2	Graphics 0
22	2	Graphics 0
23	2	Graphics 0
24	2	Graphics 0
25	2	Graphics 0
26	2	Graphics 0
27	2	Graphics 0
28	65	Jump w/WVB option
29	nn	Least significant byte of DL
30	nn	Most significant byte of DL

ATARI®

mode 6. Adding line 50 takes care of the subtitle line:

```
50 POKE DL+12,6:POKE DL+13,6
```

Now that the Display List has been modified, all that remains to complete the Display List is to shorten it to 192 scan lines long. All we really have to do is to write the 3 bytes required to end a normal Display List into memory two bytes lower than the location they already occupy. This eliminates two lines of GRAPHICS 0 and brings the scan line count to 192. If you remember, the bytes required to end a Display List are a 65, followed by the memory location of the start of the Display List in LSB, MSB order. These two bytes are conveniently stored for us at memory locations 560 and 561. Lines 60 and 70 will shorten the display and complete the modifications:

```
60 POKE DL+28,65:POKE DL+29,PEEK(560)
```

```
70 POKE DL+30,PEEK(561):? CHR$(125)
```

How It Looks

When all of the above lines have been typed in and the program has been RUN, a normal display will appear on the screen with the exception that two sections of the display are black. This is because the modified lines are operating in a GRAPHICS mode that uses a different color than the background. This can be eliminated by making both background colors the same. Add line number 80 shown below; it will reset the colors so that the screen is uniform:

```
80 SETCOLOR 2,9,0:SETCOLOR 4,9,0
```

The only things remaining for our title page are the PRINT statements to write the display. Remembering that the POSITION statement is now slightly out of kilter because of the GRAPHICS 1 and 2 mode lines, the only way to write to the screen effectively is to make a guess. The guess should be based on the screen format, and repeated, by trial and error, until text appears where it is required. We know that the display is now two lines shorter than before (we eliminated them to bring the scan line count to normal). We also know that the matched set of GRAPHICS 1 and 2 lines are considered a pair of single lines by the OS. It turns out that line four is the

start of the GRAPHICS 2 line and line ten is the GRAPHICS 1 line. The following lines print to the screen, at a position derived by trial and error, to center the text and display:

```
90 POSITION 4,4:? "STAR SLAYER"
```

```
100 POSITION 4,10:? "the ultimate":POSITION 25,10:? "space game"
```

```
110 POSITION 4,17:? "(C) 1982 - SoftSide Publications"
```

Pay particular attention to line 100. When the program is RUN, the words "SPACE GAME" are on the line following the words "THE ULTIMATE", but the POSITION statement is pointing to the same line. As mentioned before, the OS now thinks that the second line is part of the first. Line 100 could be rewritten to read:

```
100 POSITION 4,10:? "the ultimate  
space game"
```

LINES 120-170 ADD SOUND.

```
120 FOR X=1 TO 4
```

```
130 FOR Y=255 TO 100 STEP -2.5
```

```
140 SOUND 0,Y,10,8:SOUND 1,351-Y,10,8
```

```
150 NEXT Y:NEXT X
```

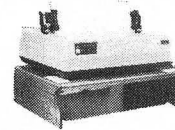
```
160 SOUND 2,201,10,8:SOUND 3,253,10,8
```

```
170 GOTO 170
```

Now that the title page is done, the screen will stay modified either until another GRAPHICS command is executed or until SYSTEM RESET is pressed. Try some experimenting with the display. Type on the modified lines in combinations of inverse and lowercase — the effects can be fascinating. Try using different codes when POKEing the Display List. Or try POKEing different locations. It's up to you. You can learn more by experimenting with your own ideas than from a hundred articles explaining how it works. If you should discover something interesting, write in and tell us about it so we can let others know. If we all pitch in for the common good and the general improvement of ATARI® software, we can all benefit.

Next time we'll get into some of the other types of Display Lists and the problems which arise when using them from BASIC. As always, I'm more than anxious to hear comments and advice from any reader of *SoftSide*. ☺

YOU'VE MADE IT,



AND IT'S BEAUTIFUL.

The Printer Stand KIT

If you want to save money *and* enjoy a fun, rewarding challenge, then the Printer Stand Kit is for you. Just check the benefits.

- The beautiful wood design is a wood worker's delight, designed to highlight your computer desk and save space for continuous paper and forms.
- All parts are precut and routed-ready for assembly to save you time.
- You choose the stain or color and finish the rich 7/8" oak sides and 3/4" oak veneer top to your preference. (Stain and varnish not included.)
- The generous size of both our stands (please compare size and price) allows the optional use as monitor or TV holder.

Instructions are complete with photos for easy assembly. Work time approx. 1½ hours.

Standard Size \$27.95*

(17 1/4" x 13" x 5 3/4") Comfortably holds the Epson MX-80, NEC-8023A-C, Okidata 82A, etc.

King Size \$31.95*

(22 3/4" x 13" x 5 3/4") Epson MX-100, Okidata 83A etc. held with ease.

ORDER TODAY FROM

Oak Kit Hardware
8689 N. 63rd Street
Brown Deer, WI 53223

*Add \$3.00 for shipping and handling. Terms: Certified check or money order. WI residents add 5% tax.

SWAT

There is an important omission from the ATARI® SWAT documentation published in Issue #30 of *SoftSide*. Before appending SWAT to a program in memory, the program to be SWATed must first be LISTed to disk or cassette (using LIST "D:FILENAME" for disk or LIST "C:" for tape). Next, turn the computer off, then on again, to clear the system and ENTER the program back into memory (using ENTER "D:filename" for disk or ENTER "C:" for tape). Because of the unique method in which ATARI® BASIC stores variables in a program, the variable table must always be in the same order to produce accurate SWAT codes. LISTing and ENTERing the program is the only known way to rebuild the variable table in a specific order so that SWAT codes can match. Note that all SWAT tables published in Issue 30 are correct and any differences in a SWAT listing indicates a typing error in your program. ☺



STAR GUN

by David Hilliard

Starbase Gunner is an arcade style game for a TRS-80® Model I or III with at least 32K RAM and disk drive. It is included as the bonus program on this month's TRS-80® Disk Version.

Congratulations! You've been promoted to *Commander* and have been assigned your own starbase. You're on your first mission in deep space and it's proving to be an "initiation by fire." Your sub-space radio screams a warning signal — an invading enemy fleet is headed straight for your station! **You** are the only one who can stop them.

Your starbase is armed with four powerful laser cannons, which are aimed directly at the four hyper-space paths the enemy will have to use. You know from intelligence reports that some of the enemy ships can be destroyed

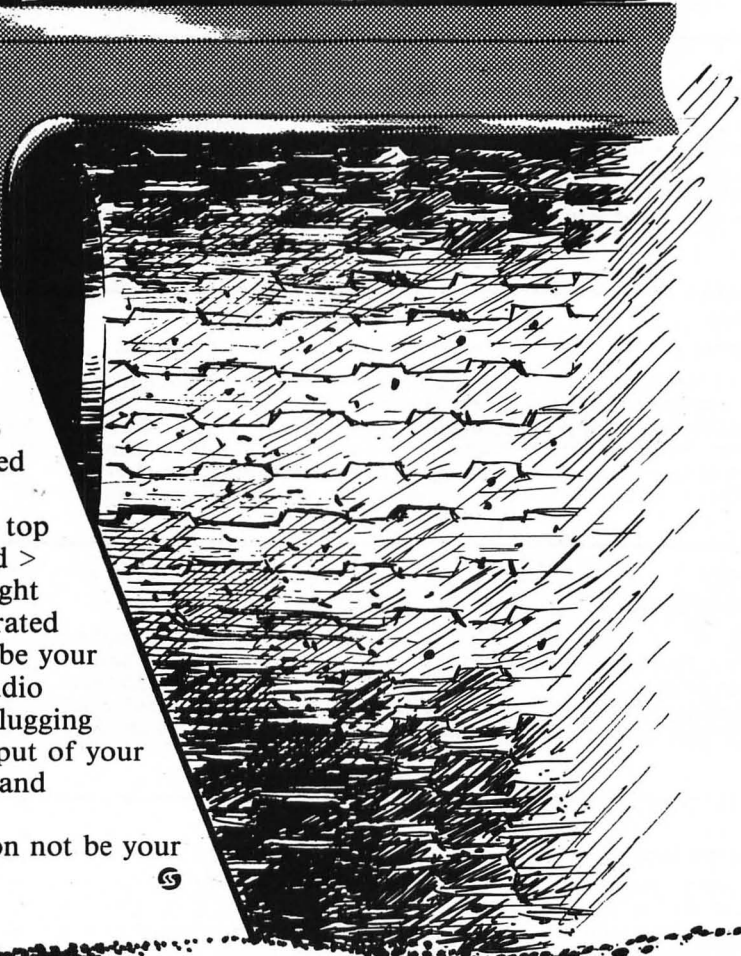
BASE NER



with a single shot while others carry strong armor shields and must be hit twice. You also know that the invaders will come out of their hyper-space warp very close to your starbase. You'll never be able to tell which path an invader will use, so you'd better be very quick on the trigger. You must also aim carefully as you will be penalized for wasted shots.

The "Q" and "W" keys fire the top left and top right lasers. The < and > keys fire the lower left and lower right lasers. Your starbase can be regenerated twice if hit. Your third chance will be your last. Your defense scanners have audio capability, which you may use by plugging the tape-out cable into the AUX input of your cassette player and pressing PLAY and RECORD.

Good luck. May your first mission not be your last.



The Adventure is



Available
on disk,
cassette or
SUPER DISK



June 1982
Arrow One Adventure

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

May 1982
Titanic Adventure

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

April 1982
Witches' Brew Adventure

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

March 1982
James Brand Adventure

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

February 1982
Klondike Adventure

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

January 1982
Windsloe Mansion Adventure

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

December 1981
Black Hole Adventure

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

November 1981
Around the World in Eighty Days Adventure

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

October 1981
Crime Adventure

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

September 1981
Jack The Ripper Adventure

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

August 1981
Treasure Island Adventure

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

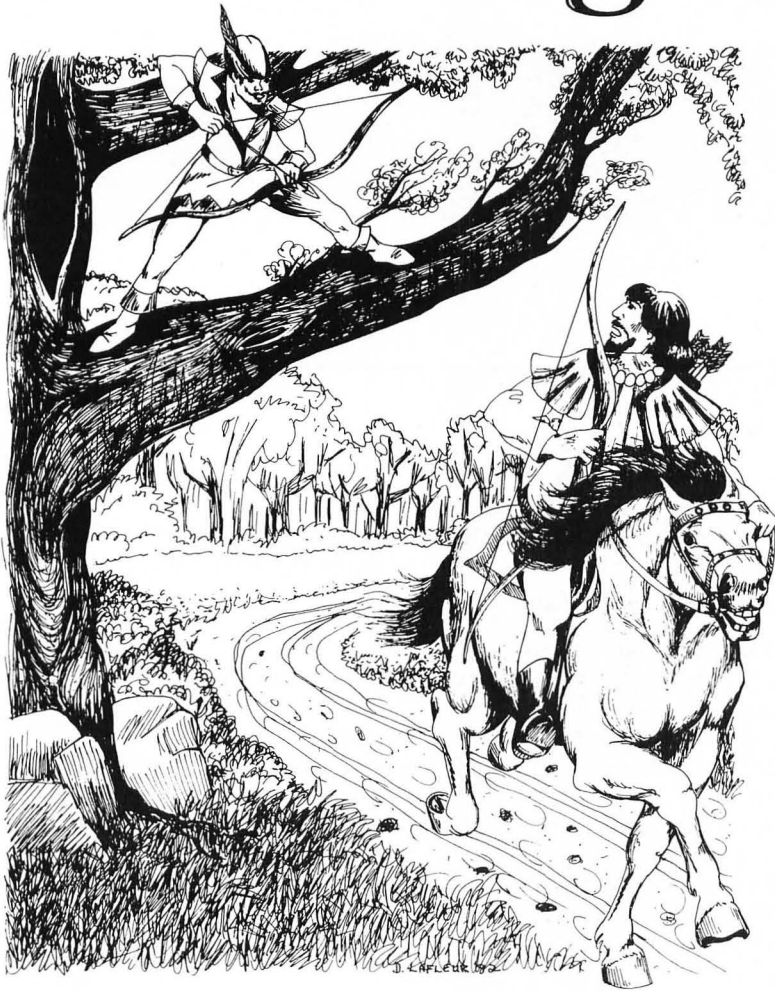
July 1981
Alien Adventure

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

June 1981
Arabian Adventure

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

Waiting for You... ♦♦♦



JULY ADVENTURE OF THE MONTH ROBIN HOOD

Thou art somewhere in Sherwood Forest...Thus it begins. Take up thy bow and arrow, and gather thy merry men, for thou art Robin Hood. There are rich merchants to rob and great deeds to do. But have care, bold Robin! The sheriff of Nottingham longs to see thy neck in a noose. Thou wilt surely need all thy wit and cunning to succeed.

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

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

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

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



6 South Street, Milford NH 03055

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

■ **Six month subscription:**

- Cassette — \$29
- Disk — \$49

■ **Individual adventures (please specify)**

- Cassette — \$7 each
- Disk — \$10 each

■ **Three adventures on one super disk (\$26 each):**

- Arabian, Alien, and Treasure Island Adventures
- Jack the Ripper, Crime, and Around the World Adventures
- Black Hole, Windsloe Mansion, and Klondike Adventures
- James Brand, Witches' Brew, & Titanic 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 _____

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.

K-Byter

Pegboard Solitaire

A TRS-80® K-Byter by Bernard Harford, Waynesboro, MS.

Pegboard Solitaire is played on a triangular board with fifteen holes containing fifteen pegs. The object of the game is to remove all the pegs but one.

Pegs are removed by jumping them with an adjacent peg into an empty hole. All three must be in a straight line.

When you start, the board will be filled with pegs. Enter the number of the hole you wish to leave empty to start the game and press ENTER. In subsequent moves, enter the number of the peg with which you are jumping and press ENTER. Then enter the number of the hole where the jumping peg will land and press ENTER. If you make a mistake in the first part of your move, press ENTER for the second part to redo. To restart the game, enter 9999 as either part of your move and press ENTER.

The puzzle can be solved in 13 moves.

```

100 CLEAR500:DEFINTF-N,T:DEFSTRA,X,O:DIMA(15),N(15):X=CHR$(179)+
CHR$(140)+CHR$(179):O=CHR$(191)+CHR$(179)+CHR$(191)
110 CLS:PRINT@3,"MOVES":;FORK=1TO15:READN(K):PRINT@N(K)-1,X;:PRI
NT@N(K)+63,K:A(K)=X:NEXT:DATA32,156,164,280,288,296,404,412,420,
428,528,536,544,552,560
120 PRINT@729,"MOVE NUMBER ";M;
130 PRINT@850,CHR$(31)"MOVE FROM ";:INPUTF
140 PRINT@867,"MOVE TO ";:INPUTT:IFF=9999ORT=9999THENRUNELSEIFF>
15ORT>15THEN170
150 IFM=0THENA(F)=0:PRINT@N(F)-1,O;:M=M+1:GOTO120
160 IF(N(F)+N(T))/2=N((F+T)/2)ANDA(F)=XANDA(T)=OANDA((F+T)/2)=XT
HEN180 ELSE170
170 PRINT@981,"INVALID MOVE - REENTER":;FORK=1TO2100:NEXT:GOTO13
0
180 A(F)=0:A(T)=X:A((F+T)/2)=0:PRINT@N(F)-1,O;:PRINT@N(T)-1,X;:P
RINT@N((F+T)/2)-1,O;:M=M+1:L=L+64:PRINT@L,USING"## ##!##";M-1;F
;:"-";T;:GOTO120
190 END
    
```

MOVES

1 13-11

2 15-13

3 5-12

4 10-8

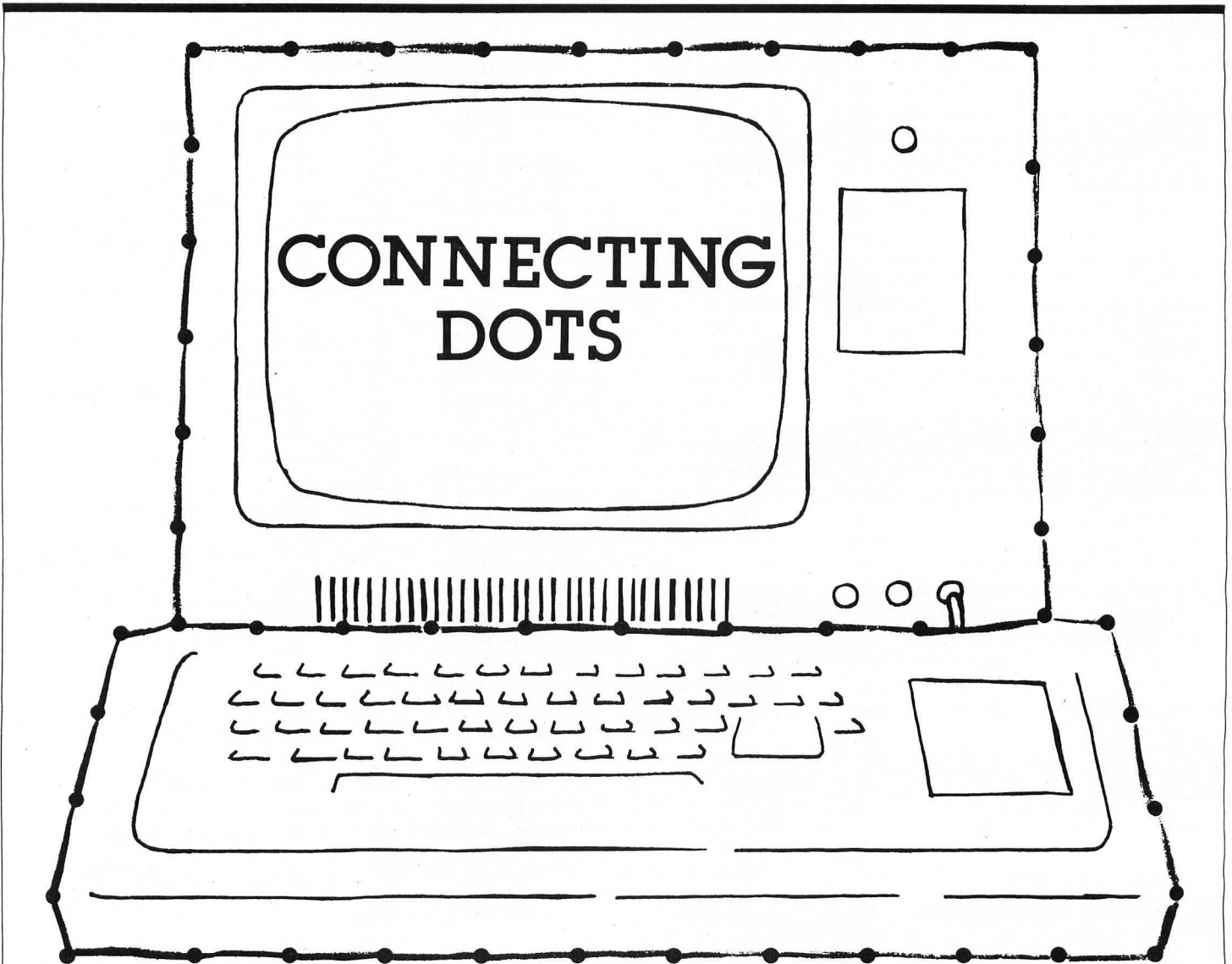
5 7-9

6 12-14

MOVE NUMBER 7

MOVE FROM ?

TRS-80®



by Charles E. Wooster

Connecting Dots is a one or two player game for a TRS-80® Model I or III with 32K RAM. (See instructions below for 16K modifications.)

In *Connecting Dots*, each player, in turn, draws a line between two adjacent dots. The line may be either horizontal or vertical. If a box is formed by the line, the player scores a point and receives another turn. When a game board has been completed, the player with the highest score wins.

When the game starts, a menu of four options is displayed. Option 1 begins the game. Option 2 allows a previously saved game to be loaded and played from the point at which

it was saved. Option 3 provides instructions on how to play the game. Option 4 allows you to quit if you decide not to play.

Option 1 will ask for the names of the players. The first player entered will be the first to move. If "TRS80" or "COMPUTER" is entered as either player, the computer will be the opponent. If both players are entered as "TRS80" and/or "COMPUTER", then, the computer will play against itself.

The computer can play with two levels of skill. The level chosen takes effect when drawing of a line will allow the other player to score. At level 1, the computer will play its best, but slowest game (10 to 360 seconds per move). At level 2, the computer may make a minor mistake or two, but will play a much

faster game (10 to 60 seconds per move).

A move is entered by typing the move coordinates NL,NL. The first NL is the Number-Letter coordinate of the "FROM" location and the second NL,NL is the Number-Letter coordinate of the "TO" location. The computer will generate the separating comma.

After the coordinates are entered, you will be asked to verify them. If they are not correct, respond "N" and you will be given another chance. If they are correct, respond "Y" or "X". If the response is "X" and a box is formed, the computer will continue forming boxes for you until no more can be formed (a time saver).

There are two special move coordinate entries available. If NL,NL is

entered as PR,NT the screen will be sent to your graphic printer. If NL,NL is entered as SA,VE the game will be saved on cassette tape for later completion or continued play. An initial menu selection of 2 will load and resume the game.

After the game has been saved, you have the option of continuing the game, starting a new game or quitting. If you choose to continue the present game, or when the game is loaded, you will be given the additional option of changing the players. If one or both of the players is "TRS80" and/or "COMPUTER" you will also be allowed to change the computer's skill level.

The program, as listed, requires at least 32K of memory and utilizes tape input and output. The instructions may be deleted from the program to make it fit into 16K (See the documentation for lines 9030-9380). Changes to provide disk input/output follow the program listing.

Lines 6400 to 6590 are the screen print routine. This routine is written for an Epson MX-80 printer

operating in MX-80 mode. A change is provided in the documentation for this routine to support the MX-80 in TRS-80 mode, or other graphics printers that support "standard" TRS-80 graphic codes. The routine uses some MX-80 features, such as double-width CHR\$(14) (CHR\$(20) to turn off), compressed CHR\$(15) (CHR\$(18) to turn off) and double-strike CHR\$(27)+CHR\$(71) (CHR\$(27)+CHR\$(72) to turn off), which may not be available or may use different codes on your printer. You may need to change this routine to suit your printer. If you don't have a printer, remove the routine by following the instructions in the routine's documentation.

The description of the game indicates there are four possible menu selections, but there are actually five. The fifth, undocumented selection provides a debugging facility for the computer playing routines. If this selection is chosen when it is the computer's turn and a scoring situation exists, timing and key variable values are displayed on the bottom three lines of the screen. Due to the use of the TIMES\$ function in this routine, debugging does not work on a Model I cassette system.

There are 2 main arrays used by the program which warrant some explanation:

The box (BX) array is a 2 dimensional array. There are 82 elements in the first dimension — 0 to 81, element 0 is not used, and elements 1 through 81 correspond to the 81 possible boxes. The second dimension contains 5 elements — 0 to 4. 0 is the number of sides completed and 1 through 4 index the lines which form the sides of the box.

The line (LI) array is a 2 dimensional array. There are 181 elements in the first dimension — 0 to 180, element 0 is not used, elements 1 through 90 correspond to the 90 possible horizontal lines and elements 91 through 180 correspond to the 90 possible vertical lines. The second dimension contains 5 elements — 0 to 4. 0 is the line number if drawn, zero if not, and 1 and 2 are the X,Y screen coordinates of the left/top of the line. 3 and 4 index the boxes formed by the line.

The computer playing routines use these variables:

C1: Index to CX array.
 C2: Index to CY array.
 C3: Index to CZ array.
 C4: Index to CL array.
 CB: Current simulated score counter.
 CC: Lowest simulated score counter.
 CL(180): Simulated lines drawn (level 1 and 2).
 CP(180): Simulated lines drawn (level 2 only).
 CX(180): Lines which will not allow score to occur.
 Simulated line array.
 Used by fireworks subroutine as a work array.
 CY(100): Lines which allow score to occur.
 Used by fireworks subroutine as a work array.
 CZ(81): Simulated box array.

The player defining variables are:

PC: Index to player arrays.
 PL(1): Computer skill level.
 PN\$(1): Name of player.
 PS(1): Player's score.
 PW: Index of winning player.
 PX: Index of losing player.

Miscellaneous variables:

BT: Total boxes formed.
 CO: Index to screen line position (save routine).
 DB: Debug switch.
 ER & ER\$: To location "L" coordinate.
 HE & HE\$: From location 'N' coordinate.
 LN: Number of last line drawn or next to be drawn.
 Lowest scoring line in computer routines.
 LT: Total lines drawn.
 RE & RE\$: From location "L" coordinate.
 RO: Index to RO\$ array.
 RO\$(12): Box, line and screen save array.
 SC: Score from last line drawn.
 TH & TH\$ - To location "N" coordinate.
 TX\$: What is next value for instruction routine.
 X\$: Keyboard response.
 X, Y & Z: Vulnerable work variable.

Ancestry I/III

TRS-80* based genealogy system
 for both the amateur and
 professional.

Features:
 Individual Records
 Family Group Sheets
 Ancestor Charts

Model I or III, 48K,
 single or dual drive,
 printer for production
 of sheets and charts.

\$69.95

+

\$2.00 shipping/handling
 (Mass. residents add 3%)

Soft-Gene
 11 John Swift Rd.
 Acton, MA 01720

*TRS-80 is TM of Tandy Corp.

TRS-80®

```

#####
$   TRS-80 BASIC   $
$   "DOTS"        $
$  AUTHOR: CHARLES WOOSTER $
$  (C) 1982  SOFTSIDE $
#####

```

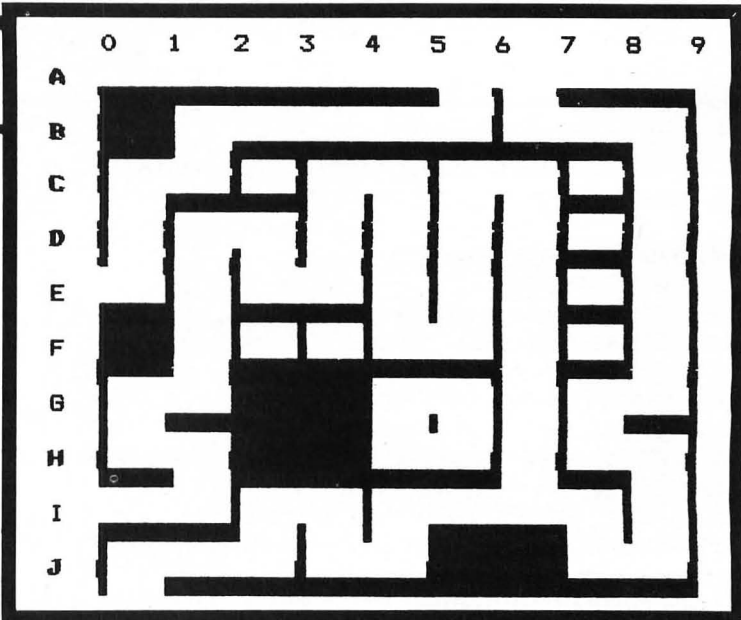
1000 GOTO8030

Entered when it is the computer's turn and a move will allow the opponent to score. The routine expects array CY to contain the index pointers into array LI for the lines not drawn which will complete the 3rd side of at least one box. The routine simulates the moves allowed by each line in array CY and picks the line which allows the fewest moves at skill level 1. At skill level 2 the routine bypasses lines which were already touched in previous moves, allowing a faster selection, but causing the routine to overlook a lower score under certain circumstances.

```

2030 IF C2>0GOTO2050
2040 PRINT@B96,CHR$(31);PN$(PC);"'s unable to locate a move - game terminated.":STOP
2050 FORX=0TO180:CL(X)=0:CP(X)=0:CX(X)=LI(X,0):NEXTX
2060 FORX=0TO81:CZ(X)=BX(X,0):NEXTX
2070 CC=81:LN=0
2080 FORX=1TOC2
2090 IF CP(CY(X))<>0ANDPL(PC)>16GOTO2360
2100 PRINT@B96+LEN(PN$(PC))+16,"lets see maybe ";
2110 PRINTCHR$(LI(CY(X),1)-22)/8+4B);CHR$(LI(CY(X),2)-10)/3+65);";";
2120 IF CY(X)<91THENY=8:Z=0ELSEY=0:Z=3
2130 PRINTCHR$(LI(CY(X),1)-22+Y)/8+4B);CHR$(LI(CY(X),2)-10+Z)/3+65);
2140 IF DB=1PRINT@960,"FROM: ";X;" TO: ";C2;" CRNT: ";CY(X);" LOW: ";CC;" LINE: ";LN;
2150 CB=0:C4=1:CL(C4)=CY(X):CP(CY(X))=CY(X):CX(CY(X))=CY(X)
2160 IFLI(CY(X),3)=0THEN C3=LI(CY(X),4):CZ(C3)=CZ(C3)+1:GOTO2180E
LSEC3=LI(CY(X),3):CZ(C3)=CZ(C3)+1
2170 IFLI(CY(X),4)<>0THEN C3=LI(CY(X),4):CZ(C3)=CZ(C3)+1
2180 IF C3<>36GOTO2270
2190 FORY=1TO4
2200 IF CX(BX(C3,Y))=0THEN C4=C4+1:CL(C4)=BX(C3,Y):CP(BX(C3,Y))=BX(C3,Y):CX(BX(C3,Y))=BX(C3,Y):Y=4
2210 NEXTY
2220 CZ(C3)=CZ(C3)+1:CB=CB+1
2230 IFLI(CL(C4),3)<>0ANDLI(CL(C4),3)<>0THEN C3=LI(CL(C4),3):CZ(C3)=CZ(C3)+1:GOTO2250
2240 IFLI(CL(C4),4)<>0ANDLI(CL(C4),4)<>0THEN C3=LI(CL(C4),4):CZ(C3)=CZ(C3)+1ELSE2260
2250 IF C3=4THEN CB=CB+1
2260 IF CB=>CCTHEN2290ELSE2180
2270 IFLI(CY(X),4)<>0ANDCZ(LI(CY(X),4))=3THEN C3=LI(CY(X),4):GOTO2190
2280 IF CB<CCTHEN CB=LN:LN=CY(X)
2290 FORY=1TOC4
2300 IFLI(CL(Y),3)<>0THEN C3=LI(CL(Y),3):CZ(C3)=CZ(C3)+1
2310 IFLI(CL(Y),4)<>0THEN C3=LI(CL(Y),4):CZ(C3)=CZ(C3)+1
2320 CX(CL(Y))=0:CL(Y)=0
2330 NEXTY

```



```

2340 IF C4=1OR C4=C2THEN X=C2
2350 IF DB=1THEN PRINT@B54,RIGHT$(TIME$,5);
2360 NEXTX
2370 IFLN<>0THEN3520ELSE2040

```

Entered when it is the computer's turn and the boxes on either side of the line last drawn cannot be completed. The routine first inspects the BX array looking for a box which can be completed. If no boxes are found to complete, it builds arrays CX and CY. Array CX indexes lines which will not allow the opponent to score, array CY indexes lines which, when drawn, will allow the opponent to score. After the arrays are built and the CX array contains entries, a random selection is made.

```

2430 IF X$="X"GOTO3040
2440 C1=0:C2=0:X=0
2450 IF DB=1THEN FORY=0TO2:POKE16919+Y,0:NEXTY:PRINT@B32,RIGHT$(TIME$,5);
2460 FORY=1TO81
2470 IF BX(Y,0)=3THEN X=Y:Y=81
2480 NEXTY:IF X<>0GOTO2660
2490 FORX=1TO180
2500 IFLI(X,0)<>0GOTO2520
2510 IF BX(LI(X,3),0)<2AND BX(LI(X,4),0)<2THEN C1=C1+1:CX(C1)=XELSE
C2=C2+1:CY(C2)=X
2520 NEXTX
2530 IF DB=1THEN PRINT@B43,CHR$(30);RIGHT$(TIME$,5);
2540 IF C1<>0THEN X=RND(C1):LN=CX(X):GOTO3520ELSE2030

```

Entered when it is the computer's turn with variable LN indexing the last line drawn. The routine inspects the BX array for boxes on either side of the line to see if either one can be completed.

```

2630 PRINT"Hummm ";
2640 IF BX(LI(LN,3),0)<>0AND BX(LI(LN,4),0)<>0GOTO2430
2650 IF BX(LI(LN,3),0)=3THEN X=LI(LN,3)ELSE X=LI(LN,4)
2660 FORY=1TO4
2670 IFLI(BX(X,Y),0)=0THEN LN=BX(X,Y):Y=4
2680 NEXTY:GOTO3520

```

Determines which player is next, displays the current score, then prompts the player for the move coordinates. Invokes the PR,NT and SA,VE options when they are chosen.

```

3030 IFPC=0THENPC=1ELSEPC=0
3040 PRINT@0,"Line";LT;" Score: ";CHR$(183)+CHR$(179)+CHR$(187
);" ";PN$(0);";";PS(0);" ";STRING$(3,191);" ";PN$(1);";";PS(1
);
3050 PRINT@896,CHR$(31);PN$(PC)'"s turn - ";
3060 IFPL(PC)<>0GOTO2630
3070 PRINT"enter move co-ordinates <NL,NL>? ";
3080 GOSUB6830:HE=X$:IFHE$=""THEN3080
3090 GOSUB6830:RE=X$:IFRE$=""THEN3090ELSEPRINT",";
3100 GOSUB6830:TH=X$:IFTH$=""THEN3100
3110 GOSUB6830:ER=X$:IFER$=""THEN3110
3120 IFHE$="P"ANDRE$="R"ANDTH$="N"ANDER$="T"THENGOSUB6430:GOTO30
40
3130 IFHE$="S"ANDRE$="A"ANDTH$="V"ANDER$="E"GOTO7030
3140 PRINT@896,CHR$(30);PN$(PC)'"s turn - verify co-ordinates ";
HE$;RE$;";";TH$;ER$;" <Y,N or X>? ";
3150 GOSUB6830:IFX$=""THEN3150
3160 IFX$="N"GOTO3050
3170 IFX$<>"Y"ANDX$<>"X"GOTO3530
3180 HE=ASC(HE$)-48:IFHE<0ORHE>96GOTO3530
3190 IFASC(RE$)<ASC("A")ORASC(RE$)>ASC("J")GOTO3530
3200 TH=ASC(TH$)-48:IFTH<0ORTH>96GOTO3530
3210 IFASC(ER$)<ASC("A")ORASC(ER$)>ASC("J")GOTO3530

```

Validates the entered move coordinates and converts them to a line number in variable LN. The routine then ensures that the specified line was not previously drawn.

```

3430 HE=22+HE#8:RE=10+(ASC(RE$)-65)*3
3440 TH=22+TH#8:ER=10+(ASC(ER$)-65)*3
3450 IFHE=THANDRE=ERGOTO3530
3460 IFHE<>THANDRE<>ERGOTO3530
3470 IFHE<>THANDABS(HE-TH)<>8GOTO3530
3480 IFRE<>ERANDABS(RE-ER)<>3GOTO3530
3490 IFHE>THTHENHE=TH:TH=HE+8
3500 IFRE>ERTHENRE=ER:ER=RE+3
3510 IFHE<>THTHENLN=((HE-22+8)/8)+(((RE-10)/3)*9)ELSELN=90+((RE-
10+3)/3)+((HE-22)/8)*9
3520 IFLI(LN,0)=0GOTO3630
3530 PRINT@896,CHR$(31);PN$(PC)'"s Boo-Boo! - ";GOTO3060

```

Draws the line specified by variable LN, updates the BX and LI arrays, and increments the LT line total variable.

```

3630 LT=LT+1:IFLN>96GOTO3650
3640 FORX=LI(LN,1)+1TOLI(LN,1)+7:SET(X,LI(LN,2)):NEXTX:GOTO3660
3650 FORX=LI(LN,2)+1TOLI(LN,2)+2:SET(LI(LN,1),X):NEXTX
3660 IFLI(LN,3)<>0THENBX(LI(LN,3),0)=BX(LI(LN,3),0)+1
3670 IFLI(LN,4)<>0THENBX(LI(LN,4),0)=BX(LI(LN,4),0)+1
3680 LI(LN,0)=LN

```

Determines if a box has been made on either side of the line specified by the LN variable and sets the SC variable accordingly. When a box has been made by the second player, the box is filled.

```

3830 SC=0
3840 IFLN>96GOTO3910
3850 IFBX(LI(LN,3),0)=4THENSC=SC+1ELSE3880
3860 IFPC=0GOTO3880
3870 FORY=LI(LN,2)-2TOLI(LN,2)-1:FORY=LI(LN,1)+1TOLI(LN,1)+7:SET
(X,Y):NEXTX,Y
3880 IFBX(LI(LN,4),0)=4THENSC=SC+1ELSE4030
3890 IFPC=0GOTO4030
3900 FORY=LI(LN,2)+1TOLI(LN,2)+2:FORY=LI(LN,1)+1TOLI(LN,1)+7:SET
(X,Y):NEXTX,Y:GOTO4030
3910 IFBX(LI(LN,3),0)=4THENSC=SC+1ELSE3940
3920 IFPC=0GOTO3940
3930 FORY=LI(LN,1)-7TOLI(LN,1)-1:FORY=LI(LN,2)+1TOLI(LN,2)+2:SET
(X,Y):NEXTX,Y
3940 IFBX(LI(LN,4),0)=4THENSC=SC+1ELSE4030
3950 IFPC=0GOTO4030
3960 FORY=LI(LN,1)+1TOLI(LN,1)+7:FORY=LI(LN,2)+1TOLI(LN,2)+2:SET
(X,Y):NEXTX,Y

```

Increments the player's score (PS array) and the BT box total variable based on the value in variable SC. Then determines if the game is complete (a total of 81 boxes are complete).

```

4030 IFSC=0THENX$="" :GOTO3030
4040 PS(PC)=PS(PC)+SC:BT=BT+SC
4050 IFBT=>81GOTO4230
4060 IFX$="X"THEN2640ELSE3040

```

Performs the end-of-game functions: saves the current playing field; generates the fireworks display; announces the winner amidst fireworks; restores the playing field, and determines if another game is desired.

continued on page 88

NEW CLASSICS SOFTWARE

Pascal-80

Phelps Gates

This friendly, easy to use version of *Standard Pascal*, as reviewed in the December 1981 *Byte*, is now even better! New version works on TRS-80 Model I and Model III, under TRS-DOS, NewDOS, NewDOS 80, DOSPlus, LDOS, and DoubleDOS. An author package allows you to create your own /CMD files without any royalty payments! Upper and lower case is fully supported. You can protect memory and call machine language programs. New extensions include SET, RESET, POINT, RND, and the UCSD Include procedure. Utilities are provided to convert to and from ASCII files. Pascal 80 now comes in a binder with an 80 page manual by George Blank.

With monitor, editor, and compiler in memory at the same time, no other Pascal is easier to learn! One college found that it could teach half again as many students on the same number of computers after switching from UCSD Pascal to Pascal 80.

Full 14 digit accuracy on all math functions, including log and trig functions, makes this a serious Pascal. Disk file handling is supported, with a mail list program included as a demonstration.

Upgrades are available for those who bought Ramware Pascal 80. Call or write for information.

Send \$101 (includes shipping) to: **New Classic Software**
239 Fox Hill Road, Box 5
Denville, NJ 07834



Credit card orders: (201) 625-8838

(PASCAL-80 does not implement variant records, pointer and window variables, or functions and procedures used as parameters.)

A feast of computing ideas.

If you work with a 6502/6809-based system, you're probably hungry for the facts and ideas that will help you understand the inner workings of your computer. You want to go beyond canned software—use your computer for more than games—learn the advanced programming techniques that enable you to get the most out of your 6502/6809 system.

MICRO, The 6502/6809 Journal, gives you page after page, month after month, of solid information to sink your teeth into. **MICRO** is the premier how-to magazine for serious users of the Apple, PET/CBM, OSI, Atari, AIM, SYM, KIM, and all 6809 based systems including the TRS-80 Color Computer. It's a resource journal internationally respected by professionals in business, industry, and education. Every issue of **MICRO** keeps you informed with up-to-the-minute data on new products and publications:

- **hardware catalog** with organized, concise description
- **software catalog** in an easy-to-use format
- **new publications** listed and annotated
- **reviews and evaluations** of significant products

And there's much more:

• **In-depth hardware tutorials** bring expert advice into your home or office.

• **Detailed discussions of programming languages** deepen and broaden your programming ability.

• **Complete program listings** enable you to increase your machine's capabilities.

• **Bibliography of 6502/6809 information** helps you to find pertinent articles in a timely manner.

• **Special monthly features** with in-depth treatment of one subject or



You'll love every byte.

YES! I want to get more from my microcomputer. Please send me _____ year(s) of MICRO at \$ _____/year. (Outside U.S. and Canada, please indicate via surface or air mail.)

Name _____

Company _____

Street _____

City _____ State _____ Zip Code _____

Check enclosed for \$ _____

Charge my credit card account

VISA MasterCard

Signature _____

Card number _____

Expiration date _____

system increase your knowledge of the field.

• **Balanced mix of machine-specific and general articles** for your everyday use as well as long-range reference needs.

• **Informative advertising** focused specifically on 6502/6809 machines keeps you abreast of latest developments.

• **Reader feedback** puts you in touch with other micro-computerists.

MICRO is the magazine you need to get the most from your own 6502/6809 system!

To order, send your check or international money order (payable to MICRO) and the order form at left, to:

Subscription Fulfillment
MICRO, Dept. MI
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Or, for your convenience, call our toll-free number:

1-800-227-1617, Ext. 564

(In California, 800-772-3545, Ext. 564)

and charge your subscription to your MasterCard or VISA. (All orders must be prepaid in U.S. dollars or charged to your MasterCard or VISA.)

SUBSCRIPTION RATES (U.S. dollars)

Yearly subscription (ISSN 027-9002) saves 20% off the single-issue price.

U.S. \$24*

Canada \$27

Europe \$27 (\$42 by air mail)

Mexico, Central America, Mideast, North and Central Africa \$27 (\$48 air)

South America, Far East, South Africa, Australasia \$27 (\$72 air)

* SPECIAL OFFER—U.S. ONLY:

Save even more—30% off single-issue price: 2 years, \$42

Dept. S S

```

4230 IFPS(0) > PS(1) THEN PW=0:PX=1 ELSE PW=1:PX=0
4240 PRINT@0,CHR$(30);:PRINT@832,CHR$(31);
4250 GOSUB5030:FORX=1TO500:NEXTX
4260 CLS:PRINT@448,TAB(22);"The winner is:"
4270 FORX=1TO250:NEXTX:GOSUB5430
4280 FORX=1TO3
4290 CLS:PRINT@448,CHR$(23);TAB((32-LEN(PN$(PW)))/2);PN$(PW)
4300 FORY=1TO250:NEXTY:GOSUB5430
4310 NEXTX
4320 CLS:PRINT@0,"The final score was: ";PN$(PW);PS(PW);"to ";PN
$(PX);PS(PX);"!!!":GOSUB5230
4330 PRINT@896,"Another game <Y or N>? ";
4340 GOSUB6830:IFX$=""THEN4340
4350 IFX$="N"GOTO9830
4360 IFX$("<")"Y"GOTO4330
4370 BT=0:LT=0:LN=0:PS(0)=0:PS(1)=0
4380 FORX=0TO180
4390 LI(X,0)=0:IFX<B2BX(X,0)=0
4400 NEXTX
4410 PRINT@896,"Same players (winner goes first) <Y or N>? ";
4420 GOSUB6830:IFX$=""THEN4420
4430 IFX$="N"GOTO8150
4440 IFX$("<")"Y"THEN4410 ELSE PC=PW:GOTO8230

```

Converts the 180 zero element values of the 2nd dimension of the LI array (LI(180,0)) to a string value in the zero element of the RO\$ array (RO\$(0)). The 81 zero element values of the 2nd dimension of the BX array (BX(81,0)) are converted to a string value in the 1st element of the RO\$ array (RO\$(1)). During these conversions 75 and 128, respectively, are added to the values to prevent cassette tape save and load errors. The current playing field screen display is then saved in the remaining 11 elements of the RO\$ array.

```

5030 FORRO=0TO12:RO$(RO)="
5040 IFRO>160TO5070ELSEIFRO=160TO5060
5050 FORX=1TO180:RO$(RO)=RO$(RO)+CHR$(LI(X,0)+75):NEXTX:GOTO5150
5060 FORX=1TO81:RO$(RO)=RO$(RO)+CHR$(BX(X,0)+128):NEXTX:GOTO5150
5070 FORCO=8TO050:X=RO$64+CD+15360:RO$(RO)=RO$(RO)+CHR$(PEEK(X))
5080 IFRO=260TO5140
5090 IFCD<100RCD>48GOTO5140
5100 IFPEEK(X)=32THENPOKEY,191:ELSEIFPEEK(X)>=128THENPOKEY,319-P
EEK(X)
5140 NEXTCO
5150 NEXTRO:RETURN

```

Reverses the effects of the playing field save subroutine (lines 5030 through 5150).

```

5230 FORRO=0TO12
5240 IFRO>160TO5270ELSEIFRO=160TO5260
5250 FORX=1TO180:LI(X,0)=ASC(MID$(RO$(RO),X,1))-75:NEXTX:GOTO5280
5260 FORX=1TO81:BX(X,0)=ASC(MID$(RO$(RO),X,1))-128:NEXTX:GOTO5280
5270 PRINT@ (RO-1)*64+64+8,RO$(RO)
5280 NEXTRO:RETURN

```

Creates the fireworks display for the end of game routine. This routine was originally

published in the April 1981 issue of SoftSide on page 24 as a one liner by Mark Nelson, Palmyra, WI.

```

5430 CLS
5440 FORY=1TO10:CY(Y)=RND(127):CY(Y)=RND(47):NEXTY
5450 FORY=1TO10
5460 SET(CX(Y),CY(Y)):SET(CX(Y),47-CY(Y)):SET(127-CX(Y),CY(Y)):S
ET(127-CX(Y),47-CY(Y))
5470 NEXTY
5480 Y=RND(2):IFY=1THEN5440
5490 FORY=1TO100:NEXTY:CLS:RETURN

```

Displays the initial 10 by 10 dot matrix on the screen and initializes the BX and LI arrays.

```

5630 FORX=22TO94STEP8
5640 FORY=10TO37STEP3
5650 SET(X,Y)
5660 IFX>86THEN5690ELSEZ=((X-22+8)/8)+(((Y-10)/3)*9):LI(Z,1)=X:L
I(Z,2)=Y
5670 IFY>34THEN5700ELSEBX(Z,1)=Z:BX(Z,2)=Z+9
5680 LI(Z+9,3)=Z:LI(Z,4)=Z
5690 IFY<37THENZ=90+((Y-10+3)/3)+(((X-22)/8)*9):LI(Z,1)=X:LI(Z,2
)=Y
5700 NEXTY,X:Z=0
5710 FORX=1TO9
5720 FORY=90+XT0162+XSTEP9
5730 Z=Z+1:BX(Z,3)=Y:BX(Z,4)=Y+9:LI(Y+9,3)=Z:LI(Y,4)=Z
5740 NEXTY
5750 LI(X,3)=0:LI(X+81,4)=0:LI(90+X,3)=0:LI(90+X+81,4)=0
5760 NEXTX:RETURN

```

Prompts for the players' names and determines if either of the players is the computer.

```

5830 FORPC=0TO1
5840 PL(PC)=0:PRINT@640+PC*128,CHR$(31);"What is the name of pla
yer #";PC+1;:INPUTPN$(PC):IFPN$(PC)=""THEN5840
5850 NEXTPC
5860 IFPN$(0)=""TRS80"ORPN$(0)=""COMPUTER"THENPC=0:GOSUB6030
5870 IFPN$(1)=""TRS80"ORPN$(1)=""COMPUTER"THENPC=1:GOSUB6030
5880 RETURN

```

Prompts for the playing level of the computer.

```

6030 PRINT@128,CHR$(31):PRINT"Levels of play are:"PRINT
6040 PRINT"1 = ";PN$(PC);" is infallible. 10-360 seconds per mov
e."
6050 PRINT"2 = ";PN$(PC);" may make minor mistakes. 10-60 second
s per move."
6060 PRINT:PRINT"NOTE: Level of play is effective during scoring
situations."
6070 PRINT:PRINT"Which level should ";PN$(PC);" use? ";
6080 GOSUB6830:IFX$=""THEN6080
6090 PL(PC)=ASC(X$)-48:IFPL(PC)<10RPL(PC)>2THEN6030ELSERETURN

```

Allows a delay for the player to read the instructions. The TX\$ variable is used to complete the prompt message.

```

6230 PRINT:PRINT"Depress any key for ";TX$;
6240 GOSUB6830:PRINT@192,CHR$(31);:RETURN

```



Prompts the player to ready the printer, then copies the screen to the printer. It is assumed that the printer is an Epson MX-80 in MX-80 mode. For an Epson in TRS-80® mode, or for any other graphics printer using "standard" TRS-80® graphics codes (see article before using these printers, however), delete line 6550. If you do not have a graphics printer, you may skip this routine, replacing it with the line: 6430 RETURN

```
6430 PRINT@960,"Ready the printer then depress any key.";
6440 GOSUB6830:POKE16425,1
6450 FORX=1TO15:LPRINT:NEXTX:LPRINTCHR$(27);CHR$(71)
6460 LPRINTTAB(24);CHR$(14);"CONNECTING DOTS"
6470 LPRINT:LPRINTTAB(36);CHR$(15);CHR$(14);"(C) 1981 by Charles
    E. Wooster"
6480 LPRINTTAB(36);CHR$(198)+STRING$(58,211)+CHR$(185)
6490 LPRINTCHR$(27);CHR$(72);FORX=1TO10:LPRINT:NEXTX
6500 PRINT@946,CHR$(31);
6510 FORRO=15360TO16320STEP64:LPRINTSTRING$(42,32);
6520 FORCO=0TO63
6530 Y=PEEK(RO+CO)
6540 IFX<32X=X+64
6550 IFX>127ANDX<192X=X+32
6560 LPRINTCHR$(X);
6570 NEXTCO:LPRINT
6580 NEXTRO:LPRINTCHR$(18);CHR$(20);CHR$(27);CHR$(72)
6590 FORX=1TO66-(PEEK(16425)-1):LPRINT:NEXTX:RETURN
```

Prompts the player for a 1 character value. A blinking cursor is provided. If the entered character is not alphanumeric, a null string is returned to the caller.

```
6830 X$="":X=PEEK(16417)*256+PEEK(16416):Y=176
6840 POKEY,Y:Z=0
6850 X$=INKEY$:IFX$<>" THEN6870ELSEZ=Z+1:IFZ<560TO6850
6860 Y=304-Y:GOTO6840
6870 IFX$<"0"OR(X$>"9"ANDX$<"A"ORX$>"Z" THENX$="":ELSEPRINTX$;
6880 RETURN
```

Saves the current playing field and the necessary variables to tape. The player is then prompted for continuing the game.
See below for disk changes to this routine.

```
7030 GOSUB5030:CLS:PRINT"Place cassette in record then depress a
ny key.":GOSUB6830
7040 PRINT:PRINT"Saving game onto tape..."
7050 PRINT#-1,BT,DB,LT,PC,PN$(0),PS(0),PL(0),PN$(1),PS(1),PL(1)
7060 FORX=0TO12:PRINT#-1,CHR$(34);RD$(X);CHR$(34):NEXTX
7070 PRINT:PRINT"Save completed...":PRINT
7080 PRINT"Continue current game (Y or N)? ";
7090 GOSUB6830:IFX$=" THEN7090
7100 IFX$<"N"ANDX$<"Y" GOTO7080
7110 IFX$="N" THEN430ELSE7280
```

Creates the playing field to initialize the BX and LI arrays, then loads the variables from tape. It then prompts the player for a change in players and computer skill level if the computer is one of the players. The screen is then restored and the game is restarted. See below for the disk changes to this routine.

```
7230 CLS:GOSUB5630:CLS:PRINT"Place cassette in play then depress
any key.":GOSUB6830
7240 PRINT:PRINT"Reloading game from tape..."
7250 INPUT#-1,BT,DB,LT,PC,PN$(0),PS(0),PL(0),PN$(1),PS(1),PL(1)
7260 FORX=0TO12:INPUT#-1,RD$(X):NEXTX
7270 PRINT:PRINT"Reload completed..."
7280 PRINT:PRINT"Players are: ";PN$(0);" and ";PN$(1)
7290 PRINT"Do you want to change the players (Y or N)? ";
7300 GOSUB6830:IFX$=" THEN7300
7310 IFX$="N" GOTO7330
7320 IFX$<"Y" THEN7290ELSEGOSUB5830:GOTO7410
7330 IFPC=0 THENPC=1ELSEPC=0
7340 IFPL(PC)=0 GOTO7400
7350 PRINT:PRINTPN$(PC);" is playing at level ";PL(PC);"."
7360 PRINT"Do you want to change the level (Y or N)? ";
7370 GOSUB6830:IFX$=" THEN7370
7380 IFX$="N" GOTO7400
7390 IFX$<"Y" THEN7360ELSEGOSUB6030
7400 IFPC=0 THENPC=1ELSEPC=0
7410 CLS:GOSUB5230:GOTO3040
```

Displays the initial menu and prompts the player for a selection.

```
8030 CLS:CLEAR1000:DEFINTA-Z:RANDOM
8040 DIMBX(81,4),CL(180),CP(180),CX(180),CY(100),CZ(81),LI(180,4
),PL(1),PN$(1),PS(1),RD$(12)
8050 PRINT@0,TAB(24);"CONNECTING DOTS"
8060 PRINTTAB(16);"(C) 1981 by Charles E. Wooster"
8070 PRINT@192,"Menu: 1 Start game."
8080 PRINT"    2 Resume a previously saved game."
8090 PRINT"    3 Instructions."
8100 PRINT"    4 If you are a poor sport."
8110 PRINT:PRINT"What is your menu selection? ";
8120 GOSUB6830:IFX$=" THEN8120ELSEPRINT
8130 X=ASC(X$)-48:IFX<10R>5 THEN8070ELSEONXGOTO8150,7230,9030,98
30,8140
8140 DB=1
8150 GOSUB5830:PC=0
```

Sets up the initial game screen display.

```
8230 CLS
8240 FORX=0TO9
8250 PRINT@134+4*(X+1),X;:PRINT@200+X*64,CHR$(65+X);
8260 NEXTX:GOSUB5630:GOTO3040
```

Displays the instructions and returns to the menu. For 16K tape systems, you must not type this routine. It will not fit. In its place, put the following single line:
9030 GOTO8070

```
9030 PRINT@192,CHR$(31);"The game of Connecting Dots is a two
player game. Each player"
9040 PRINT"in turn draws a line between two adjacent dots. The l
ine may be"
9050 PRINT"either horizontal or vertical. If a box is formed by
the line,"
9060 PRINT"the player scores a point and receives another tu
rn. When a"
9070 PRINT"game board has been completed the player with the hig
hest score"
9080 PRINT"wins."
```

```

9090 TX$="more instructions.":GOSUB6230
9100 PRINT"A move is entered by typing the move co-ordinates NL
,NL where"
9110 PRINT"the first NL is the number-letter combination that
forms the"
9120 PRINT"'FROM' location and the second NL is the number-lette
r combin-"
9130 PRINT"ation that forms the 'TO' location. The computer will
generate"
9140 PRINT"the separating comma. If NL,NL is PR,NT the scree
n will be"
9150 PRINT"sent to the printer. If NL,NL is SA,VE the game will
be saved"
9160 PRINT"on cassette tape for later completion. A menu selec
tion of 2"
9170 PRINT"will resume a game which was saved in this manner."
9180 GOSUB6230
9190 PRINT"After the co-ordinates are entered they must be ve
rified. If"
9200 PRINT"they are not correct respond 'N' and you will be giv
en another"
9210 PRINT"chance. If they are correct respond 'Y' or 'X'. If th
e response"
9220 PRINT"is 'X' and a box is formed the computer will contin
ue forming"
9230 PRINT"boxes for you until no more can be formed (a time sav
er)."
```

```

9240 GOSUB6230
9250 PRINT"If 'TRS80' or 'COMPUTER' is entered as either pla
yer that"
9260 PRINT"player will be the computer. If both players are e
ntered as"
9270 PRINT"'TRS80' and/or 'COMPUTER', then the computer will
play with"
9280 PRINT"itself.":PRINT
9290 PRINT"There are 2 levels of play when the computer is one o
r both of"
9300 PRINT"the players. The level specified takes affect when
the game"
9310 PRINT"has progressed to the point where drawing a line w
ill allow"
9320 PRINT"the other player to score. At level 1 the computer
will play"
9330 PRINT"its best but slowest game. At level 2 the computer m
ay make a"
9340 PRINT"minor mistake or two but will play a faster game."
9350 TX$="good luck.":GOSUB6230
9360 PRINT"TRS80 wishes you the best of luck and may the better
computer"
9370 PRINT"win!!!!!"
9380 TX$="menu selection.":GOSUB6230:GOTO8070

```

Says "Bye-Bye" and ends the program.

```

9830 CLS:PRINT@448,TAB(23);CHR$(23);"Bye-Bye"
9840 FORX=1TO500:NEXTX:CLS:CLEAR50:END

```

If you have a disk system, the following lines should be added or substituted for the corresponding lines in the tape version above.

```

7030 GOSUB5030
7032 CLS:PRINT"Image number (1-9)";:GOSUB6830
7034 IFX$<"1"ORX$>"9"THEN7032
7036 OPEN"D",1,"DOTSAVE"+X$+"/DAT"
7040 PRINT:PRINT"Saving game onto disk..."
7050 PRINT#1,BT;";";DB;";";LT;";";PC;";";PN$(0);";";PS(0);";";PL
(0);";";PN$(1);";";PS(1);";";PL(1)
7060 FORX=0TO12:PRINT#1,CHR$(34);RO$(X);CHR$(34):NEXTX:CLOSE1
7230 CLS:GOSUB5630
7232 CLS:PRINT"Image number (1-9)";:GOSUB6830
7234 IFX$<"1"ORX$>"9"THEN7232
7236 OPEN"I",1,"DOTSAVE"+X$+"/DAT"
7240 PRINT:PRINT"Reloading game from disk..."
7250 INPUT#1,BT,DB,LT,PC,PN$(0),PL(0),PN$(1),PS(1),PL(1)
7260 FORX=0TO12:INPUT#1,RO$(X):NEXTX:CLOSE1

```

TRS-80® SWAT TABLE FOR: DOTS

LINES	SWAT CODE	LENGTH
1000 - 2130	HQ	427
2140 - 2230	DI	507
2240 - 2350	BY	402
2360 - 2520	EN	276
2530 - 3060	AM	425
3070 - 3180	NG	451
3190 - 3510	YT	415
3520 - 3860	FP	380
3870 - 4040	CA	475
4050 - 4320	SU	352
4330 - 4440	MM	305
5030 - 5240	EI	349
5250 - 5630	SV	340
5640 - 5750	EQ	380
5760 - 6070	JK	520
6080 - 6500	JZ	403
6510 - 6850	JB	279
6860 - 7110	UK	412
7230 - 7340	MU	428
7350 - 8070	RE	419
8080 - 8260	ID	340
9030 - 9110	LI	554
9120 - 9200	MH	560
9210 - 9290	HG	515
9300 - 9830	BG	520
9840 - 9840	XR	24

ZBASIC 2.0

A review by Tim Knight

by Andy Gariepy (Simutek, Inc.).
System Requirements: 16K TRS-80® Model I or III. Suggested retail price: Disk — \$89.95, Tape — \$79.95

What is a compiler?

A compiler, in simple terms, is an advanced program which takes a computer program in one language and changes it into an equivalent program in another language. The most popular type of compiler for the TRS-80® is a BASIC compiler, one which takes another BASIC program and "transforms it" into Machine Language.

The most obvious advantage of this process is an improvement in execution speed. This improvement may very well enhance your chances of selling a program you have written to the commercial market. For instance, a BASIC game doesn't have a great chance of making it big on the software market. However, compiled into Machine Language, the extra speed it gains may give it a much better shot at selling. Machine Language programs execute approximately 300 times faster than BASIC programs, and compiled programs (BASIC transformed into Machine Language) execute about 100 times as fast as their BASIC counterparts.

There are many compilers for the TRS-80® on the market today. There is *BASCOS*, by Microsoft, but it is too expensive for most, only for disk systems, and requires payment of substantial royalties to Microsoft if a program made with it is sold. There are other, less expensive compilers like *Tiny Comp* and *Accel 2*, but these aren't as powerful as *BASCOS* and don't offer many speed improvements. A while ago, a program called *ZBASIC* was introduced. It wasn't a phenomenal compiler either, considering that many restrictions were placed on programs written with it. However, something new and exciting has come out from the makers of the original *ZBASIC* (Simutek, Inc.).

New and Improved

ZBASIC 2.0 is a completely new compiler for the TRS-80®. It will work on a Model I or III disk or tape system with 16K or more memory. There are no royalties imposed on programs made with *ZBASIC 2.0* and it is such an exciting and innovative program that it should prove quite popular.

ZBASIC 2.0 on tape contains six versions of the program and is accompanied by a nearly 100 page manual. The manual explains that

"The manual alone is very exciting. The features described are so incredible and varied — I can't believe so much can be stored in one computer. Extensive instructions are given on how to load any single version of ZBASIC for either disk or tape."

ZBASIC 2.0, by Andy Gariepy, is an interactive compiler which will reside in memory along with any BASIC programs. The user needs merely to type in a BASIC program, press three keys at the same time, and the program will be compiled and ready to test. This is a great convenience.

The manual alone is very exciting. The features described are so incredible and varied — I can't believe so much can be stored in one computer. Extensive instructions are given on how to load any single version of *ZBASIC* for either disk or tape.

I loaded the program and started to experiment. Initially, the pro-

gram asks for configuration standards. These standards set parameters such as the string memory size which should be allocated and what the maximum string size will be. *ZBASIC* has a unique method for storing variables, so these questions may be either ignored or set to the user's preference.

After answering these questions, I checked in the manual to see what I could do next. There is a short program, written in BASIC, which uses the SET (X,Y) command to make a white box on the screen. I entered the program and ran it in BASIC. As could be expected, it ran quite slowly. I then pressed the compiling keys, the comma, the period, and the division sign (./) all at the same time to compile the program. It compiled instantly, a nice change from the slow *Accel 2* program compiling to which I've grown so accustomed.

Unbelievable!

I ran the compiled program and couldn't believe my eyes! Almost immediately the screen was filled with a white box, quite a contrast from its slow BASIC counterpart. I then had three options. I could run the program again, return to BASIC, or save the program to tape. Any of these may be executed with a single key and are a tremendous improvement over the *ACCEL 2* method of loading a separate program and specifying memory addresses galore before anything can be accomplished.

There were a couple of things I discovered immediately — first, only integer math is supported by *ZBASIC 2.0* using the standard methods. (I later found I could have ultra-high precision math using a special *ZBASIC* technique that I'll describe below.) Also, a *ZBASIC 2.0* compiled program cannot be stopped by using the BREAK key. Instead, a key-checking device must

be put in the program to test for the BREAK key being pressed and stop the program.

Differences

Most of BASIC's commands work well with ZBASIC. However, there are some differences — almost all positive. These differences are really just improvements on BASIC itself. For instance, the POS command will not return a number between 0 and 63 (which is fairly useless) but a number between 0 and 1023 instead, specifically showing the position of the cursor on the screen (very useful!). Another important difference is that STOP will end the program and END will reboot the computer (the equivalent of hitting the reset key).

There are extensive sections in the manual describing the minor differences in features such as FOR/NEXT loops, READ/DATA operations, and INPUT. The manual is generally well detailed in pointing out these differences. Tape input/output and Disk input/output are also well documented.

Precisely!

High precision math is one of the greatest features of ZBASIC 2.0. It is termed as "@ MATH" since the @ symbol signifies where in the program a high precision math operation is taking place. Believe it or not, the high precision math package is accurate up to sixty-two places! There are a few restrictions on "@ MATH." Only one operation is permitted per line and numbers must be defined as strings rather than integers. However, sixty-two place precision is certainly worth the extra consideration.

Special Functions

There are many special features and functions in ZBASIC 2.0, making it a superior compiler. The special commands are not recognized by BASIC, so the programs using them can't be tested in BASIC before compiling (though you may alternate between the compiled program and BASIC whenever you like).

The block move command,

similar to the LDIR and LDDR instructions in Assembly Language, is implemented in ZBASIC 2.0 with the SIN and COS commands. (You can see how the results would be very different if this were tested in BASIC.) The block move is implemented at 100,000 bytes per second which means that every single byte of the entire computer's memory could be moved in less than a second. Some useful purposes for block move commands could be to examine large areas of memory, to exchange certain areas of memory, or to relocate other programs. The possibilities are exciting.

Next is the compare command. This is an extremely rapid search through memory for any type of argument. It is implemented with the CINT and CDBL commands. Some very handy applications for this might be to locate every GOTO statement in memory, or perhaps to find the exact memory location of the Tandy Copyright!

If one wants the entire screen to be graphically inverted, this is also supported by ZBASIC 2.0. For instance, if random graphic blocks were set in various places around the screen and the invert command was implemented (FIX), then every previously filled location would be blank, and every previously blank space would be filled with white. This can be a lot of fun in games. For instance, if you were programming a game in which an explosion was to occur, the invert (FIX) command could create some quick flashes, really looking like an explosion.

Tone generation, something nearly impossible to accomplish in regular BASIC, is supported with yet another command in ZBASIC. The tone command (CVD) will generate any tone for any length. It can be used for sound effects, music, or prompts.

In addition to these remarkable functions, features such as high speed multiplication, high speed division and high speed multiplication by 10 are fully implemented. Also, tape and disk input/output have their own special commands to enable high speed saving or retrieving of data. Yet another function is a Machine Language call (somewhat like the USR function) which is

called by the ZBASIC 2.0 command TAN.

For Machine Language programmers, ZBASIC has several other special functions. The LOC command can read the stack pointer (SP) or disable interrupts (DI) directly from BASIC. A particularly useful feature is the 16-bit peek command (CSNG). This might be used for finding out the setting of the USR bytes (bytes 16526 & 16527) without having to make two peeks. User defined functions are also implemented. Making calls to DOS is easy with the commands supported by ZBASIC 2.0 in this manner.

Handy Helpers

The appendices to the manual are very helpful. Appendix A contains a bevy of programs which won't work well or at all in regular BASIC, but are extremely powerful in ZBASIC 2.0. Graphic, Memory Move, Ping Pong, and other well-documented programs, twenty in all, are in the appendix and are a great deal of fun to experiment with.

In appendix B, the error messages are explained. Some very helpful memory maps are to be found in Appendix C. Relocation is covered in Appendix D, and an explanation of the free MISOSYS command file on the disk version follows. There is an index in the back of the manual for quick reference.

Fantastic!

As you can see, this is a program like no other. A compiler such as this can greatly expand the marketing potential of programmers not quite accustomed to Machine Language (or lacking the time to learn it) and will greatly speed up any programs compatible with ZBASIC 2.0. Above all, it is enjoyable to use, and the extensive commands are a constant source of amazement.

I highly recommend this program for any TRS-80® user. It is, without a shadow of a doubt, the most useful program I own. It is well conceived and thoroughly documented. ZBASIC 3.0 should be coming out in about a year and is rumored to make Microsoft BASIC obsolete. Bill Gates, watch out! ☺



Boston

Minneapolis

Houston

San Francisco



The Most Spectacular Extravaganza Ever... For Apple Users

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

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

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

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

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

Ticket & Hotel Information

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

Exhibitor Information

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

Applefest/Boston

Fri-Sun May 14-16, 1982

Hynes Auditorium

Show Hours: 11AM to 6PM Daily

Admission: \$6 per day or \$10 for 2 days,
\$15 for 3 days

Applefest/Minneapolis

Thurs-Sun Sept 16-19, 1982

Minnesota Auditorium and Convention Hall

Show Hours: 11 AM to 6 PM Daily

Admission: \$5 per day or \$8 for 2 days,
\$12 for 3 days, \$15 for 4 days

Applefest/Houston

Fri-Sun Nov 19-21, 1982

Albert Thomas Convention Center

Show Hours: 1PM to 10PM Daily

Admission: \$5 per day or \$8 for 2 days, \$12 for 3 days

Applefest/San Francisco

Fri-Sun Dec 3-5, 1982

Moscone Center

Show Hours: 1PM to 10PM Daily

Admission: \$5 per day or \$8 for 2 days, \$12 for 3 days

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

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

NEW PRODUCTS



FUTURE SOLUTIONS, INC.
207 E. 85th Street
Suite 447
New York, NY 10028

The **MICROCART**® is a new concept in computer cabinetry designed by computer professionals to provide both home and office users with an organized means of consolidating microsystem components. Its many features include mobility, work space expansion and flexibility, and a slide-out computer work surface which allows easy access and provides for glide-in storage when the computer is not in use. It also offers a security lock for the cabinet, insuring safety for your computerware, programs-in-process and system. Below the cabinet there is ample shelf space for the storage of printers, diskettes, printout paper, or other needs.

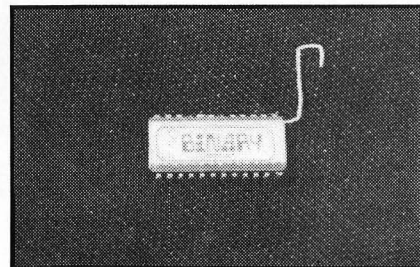
MICROCART® is available in walnut, oak and teak finishes and is 36 inches high, 32 inches wide and 22 inches deep. It retails for \$439.95.



DON'T ASK COMPUTER SOFTWARE
2265 Westwood Blvd. B-150
Los Angeles, Ca 90064
(213)397-8811

Here's a word game that's competitive, fast-paced, exciting, addictive and educational: **WORDRACE** by Don't Ask. **WORDRACE** is for a 32K ATARI® 400/800 with disk drive and BASIC Cartridge or 48K Apple™ II or II+ with disk drive and Applesoft. This one to four player word game, for players age nine to adult, features 3 levels of play and requires strategy and

speed. **WORDRACE** builds players' vocabularies while they have fun. Each turn begins with a word, six possible definitions, and a counter starting at 600 points. When you select the correct definition you score the number of points remaining. Guess incorrectly and you lose the number of points left. The disk contains over 2000 words and definitions. Don't Ask plans to issue companion disks, some containing extra words and definitions, and others with the materials to adapt the basic **WORDRACE** game to other subjects, such as famous names in history. **WORDRACE** retails for \$24.95.



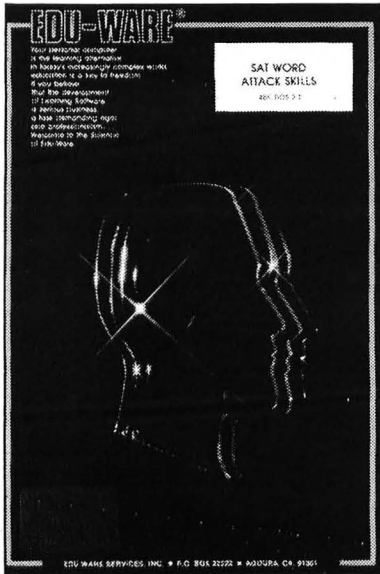
BINARY CORPORATION
3237 Woodward Ave.
Berkley, MI 48072
(313)548-0533

Now owners of ATARI® 810 disk drives, purchased prior to ATARI®'s fast formatting engineering change, can increase the formatting speed of their machines by more than 30%. ATARI® 810 disk drives with the new fast formatting change can increase in efficiency by as much as 10 percent.

FAST-CHIP is a well documented kit requiring only one lead to be wired to the disk's main electronics board. Any person who knows how to solder can accomplish this task in less than 15 minutes with a low wattage soldering iron and a screw driver.

FAST-CHIP is available at ATARI® dealers worldwide and carries a one-year guarantee. Installation is available from most local dealers for a slight fee, or the drive can be sent post paid to BINARY at the address above with payment of \$39.95 plus \$15.00 to cover shipping, handling, and installation. Retail price is \$39.95.

NEW PRODUCTS



EDU-WARE SERVICES, INC.
P.O. Box 22222
Agoura, CA 91301

College-bound students preparing for the Scholastic Achievement Test (SAT) will find two new software tutorials helpful in mastering vocabulary and deciphering new words — necessary skills for the “antonyms” portion of the test.

PSAT Word Attack Skills and *SAT Word Attack Skills* each contain two diskettes. Words, selected for their frequent appearance on the PSAT/SAT, are grouped into lessons. In each lesson, students first select the word’s synonym. The word is then reviewed and, finally, a timed test prepares students to function under pressure.

PSAT and *SAT Word Attack Skills* are independent tutorials, but may be purchased and used together as a comprehensive sequence. They’re available for the Apple II™ and II+ computer with 48K of memory and DOS 3.3. Each two-disk package retails for \$49.00.



subLOGIC COMMUNICATIONS CORP.
13 Edgebrook Drive
Champaign, IL 61820
(217)359-8482

A2-ED1 Whole Brain Spelling, written and designed by David Manton and Susan Campanini, is the first in a series of educational software packages from subLOGIC. The program has been designed to help improve spelling in a manner as entertaining as it is educationally sound. It effectively utilizes the graphics and color capabilities of the Apple™ II or II+ computer to provide positive user-feedback and to emphasize visual aspects of the learning process.

The program is user friendly. You can move to any desired lesson section, choose your own word lists to study, and proceed at your own rate. A main Spelling Menu is accessible from any portion

of the programs and lesson instructions are always available.

200 10-word lists of practice spelling words, organized in order of increasing spelling difficulty, are included with *Whole Brain Spelling*. Supplementary word lists are also available in the following categories: Medical, Scientific, Secretarial, Fairy Tale, and A Child’s Garden of Words.

A2-ED1 Whole Brain Spelling is available at most computer stores or can be ordered directly from subLOGIC. The program requires 48K memory, a disk drive and either an Apple™ II+ or an Apple™ II with Applesoft in ROM. A color monitor is recommended. For direct orders add \$1.50 for shipping and specify UPS or first class mail. Illinois residents add 5% sales tax. Unless otherwise requested the MAIN word list will be included with each program ordered. Retail price is \$34.95.

ARGUS, INC.
Box 9777
Baltimore, MD 21204
(301)321-8451

LEGGSTM (U.S. patent pending) solve the problem of where to put the paper for your Epson MX-80 printer — underneath it! *LEGGs* are clear, acrylic plastic and install in seconds with a twist of the wrist — no drilling or tools required. They provide room for 3” of paper with access from all four sides.

Available from your local computer dealer or direct from Argus, Inc. postpaid. Retail price is \$15.00.



SoftSide’s New Products section is an effort to inform our readers of newly available microcomputer products. However, *SoftSide* assumes no responsibility for product quality, company reliability or data accuracy. The information printed is submitted by promoters and selected for publication on a first-come, editorial preference, and computer compatible basis. Due to space limitations, we reserve the right to modify content of submissions. Submissions may be sent to:

SoftSide Magazine
New Products Editor
6 South Street
Milford, NH 03055

NOT VERY FAR AWAY.
IN THE NOT TOO DISTANT FUTURE —
ISSUE #32 OF
SoftSide

Advertiser's Index

Alien Group	1
Avante Garde Creations	57
Binary Computer Software	74
Classroom Computer News	43
Computer Trader®	10
Computer's Voice	71
Datamost	33
EdCom '82	5
Howard Sams	Cover III
Irv Brechner Enterprises	52
Leading Edge	Cover IV
Micro 6502/6809	87
National Computer Shows	54
New Classics Software	86
Northeast Expositions	93
Oak Kit Hardware	77
Educational Software, Inc	27
Silicon Valley Systems	40
Synapse Software	Cover II
Soft-Gene	84
Spectrum Computers	67
Tara Computer Products	63
T.H.E.S.I.S	56
Vervan Software	15

SoftSide Publications	
The Adventure of the Month	80, 81
Attention Authors	39
Back Issues	30, 31
Binders	51
SoftSide	63
Translation Contest	9

ADVENTURE/SIMULATION

DARE YOU ATTEMPT

OPERATION SABOTAGE

... Can you destroy the alien space station and remove the secret plans.

ARE YOU READY TO EXPLORE THE WORLD OF...

TRON in ENTERTAINMENT TOMORROW

PUT ON YOUR SURGICAL GLOVES, TURN ON YOUR COMPUTER AND DISCOVER THE...

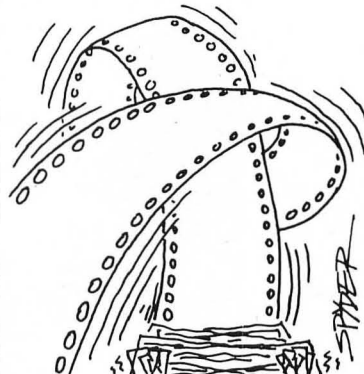
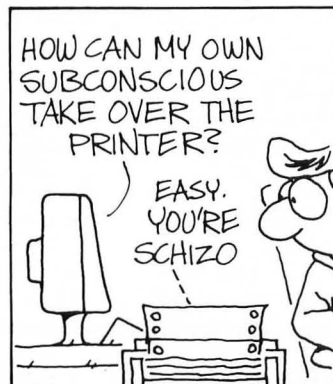
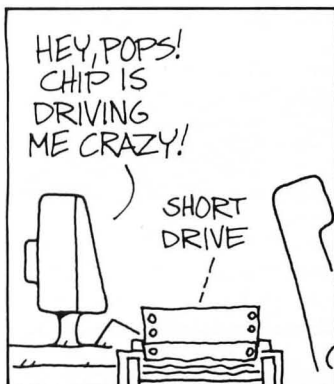
ANATOMY OF AN ADVENTURE

★ PLUS

CATS (Part Two), A report on the NCC, another Apple Diskourse, Reviews and more fun than a barrel of Orcs.

MACHINE HEAD

BY SPYDER



Only Mitch Waite can take the basic building blocks of microcomputer knowledge and present them to you in such a friendly and enjoyable way. Get down to basics with Mitch Waite's PRIMER SERIES from Sams.

BASIC PROGRAMMING PRIMER, No. 21586, \$11.95. A complete guide to BASIC, today's most widely used microcomputer language. By Mitch Waite and Michael Pardee.

COMPUTER GRAPHICS PRIMER, No. 21650, \$14.95. This masterpiece by Mitch Waite shows you how to create graphics on your microcomputer video screen, including colorful drawings, plans, maps and schematics.

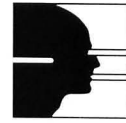
CP/M® PRIMER, No. 21791, \$14.95. A non-technical discussion of the celebrated CP/M disk operating system for 8080 and Z-80 based systems. By Mitch Waite and Stephan Murtha.

MICROCOMPUTER PRIMER (SECOND EDITION), No. 21653, \$14.50. This book is indispensable to the reader who wants to gain a solid understanding of microcomputer systems, from basic hardware through operating systems and user programming techniques. By Mitch Waite and Michael Pardee.

PASCAL PRIMER, No. 21793, \$16.95. A complete, comprehensive introduction to one of today's hottest computer languages. By Mitch Waite and David Fox.

Learn the basics of microcomputers with Sams microcomputer books. With Mitch Waite's PRIMER SERIES, it's easy as ABC.

To order these Sams books or to get the name of your local Sams retailer, call **800-428-3696 toll-free**, or **317-298-5566** and reference **AD164**.



SAMS BOOKS

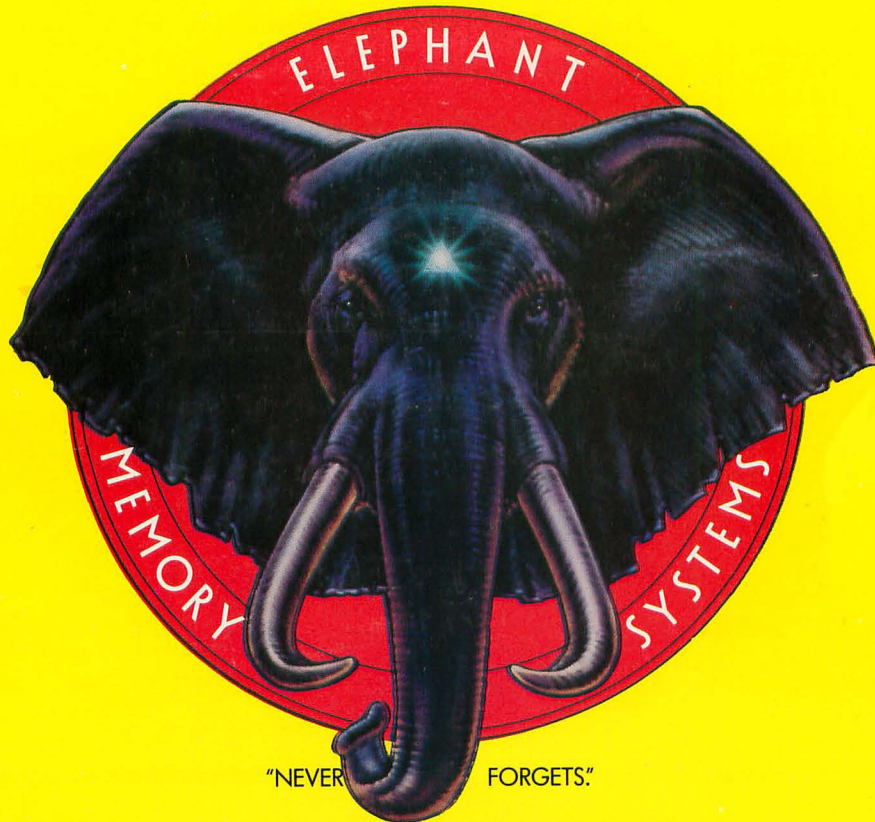
Howard W. Sams & Co., Inc.
4300 West 62nd Street, P.O. Box 7092
Indianapolis, IN 46206

EASY AS ABC



Offer good in U.S. only and expires 8/31/82. Prices subject to change without notice.

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to

make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.