

MYR1

2522

TO

93065 JOHN SBO9
BYRCN JCHNSDN
356 LAGUNA TERR
SIMI VALLEY, CA 93065

Second Class Postage Paid at Menlo Park, CA
Address Correction Requested



DR. DOBB'S JOURNAL of COMPUTER Calisthenics &

Orthodontia

Dr. Dobb's Journal is a highly respected reference journal which fills a unique and solid niche within the microcomputing world. We publish discussions and examples of general purpose system tools, articles on legislation or trends affecting computerists, a consumer watchdog column, as well as a variety of monthly columns designed to guide and help readers wend their way through the confusion of a new and burgeoning industry.

Recent issues have included: • Growing, pruning and climbing binary trees with tiny-c • Selecting business software for microcomputers • An interactive timeshared operating system for the 8080 • A critical look at the MC68000 • A Z80-ZAP disassembler • Converting 8-bit memories to 16-bits • OSI BASIC for the KIM-1 • Let your computer speak ASCII

Postage-paid
subscription card
inside this
issue

OUR READERS SAY:

"Your level of information is fantastic... the best!"

"Dr. Dobb's has presented some of the best and most intelligent software for the advanced hobbyist. Please, please stay that way!"

"Keep up the heavy software, non-commercial, short miscellaneous, unslick, lots of letters approach: it's unique among *Byte*, *Interface Age*, *Kilobaud*, etc."



Recreational COMPUTING

FORMERLY PEOPLE'S COMPUTERS

In Canada \$2.50 \$2.00

VOL 8

NO 2

ISSUE 41

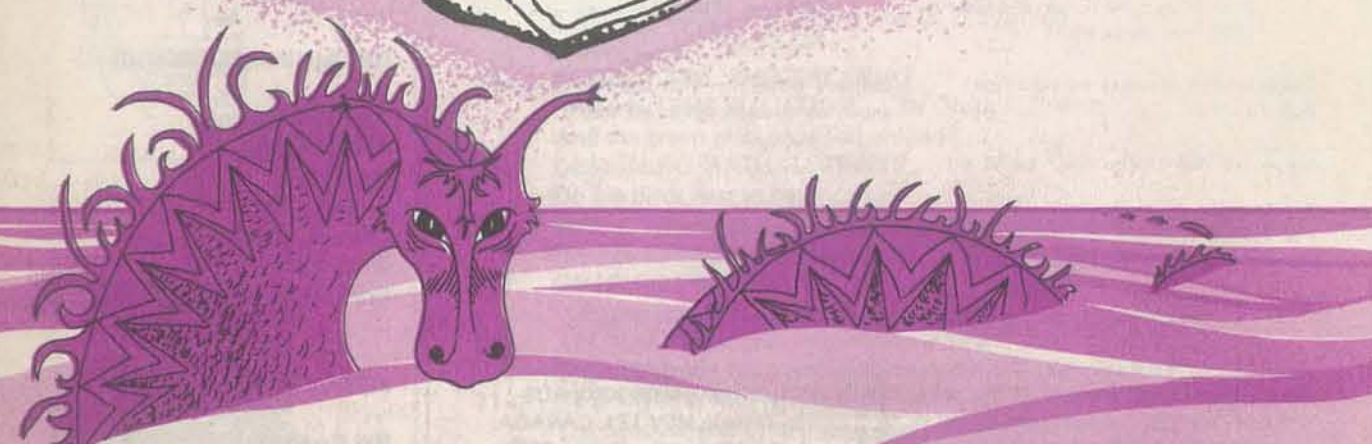
SEPT-OCT

1979

COLOR · SOUND · GRAPHICS

Texas Instruments

Atari



CRYPTARITHMS

TRS-80: OUTSIDE CONNECTION

ARCHITECTURE OF MULTI-PLAYER GAMES

SUBMITTING ITEMS FOR PUBLICATION

LABEL everything with your name, address and the *date*; tapes should also include the program name, language and system. TYPE text if at all possible, double-spaced, on 8 1/2 x 11 inch white paper. DRAWINGS should be as clear and neat as possible in black ink on white paper.

LISTINGS are hard to reproduce clearly, so please note:

- Use a new ribbon on plain white paper when making a listing; we prefer roll paper or fan-fold paper.
- Send copies of one or more RUNs of your program, to verify that it runs and to provide a sense of how things work—and to motivate more of us to read the code. RUNs should illustrate the main purpose and operation of your program as clearly as possible. Bells, whistles and special features should just be described in the documentation unless they're particularly relevant.
- Make sure your code is well documented—use a separate sheet of paper. Refer to portions of code by line number or label or address, please, not by page number. When writing documentation, keep in mind that readers will include beginners and people who may be relatively inexperienced with the language you're using. Helpful documentation/annotation can make your code useful to more people. Documentation should discuss just which cases are covered and which aren't.
- If you send us a program to publish, we reserve the right to annotate it (don't worry, we won't publish it if we don't like it).
- Last but not least, please try to limit the width of your listings: 50-60 characters is ideal. Narrow widths mean less reduction, better readability and better use of space.

LETTERS are always welcome; we assume it's OK to publish them unless you ask us not to. Upon request we will withhold your name from a published letter, but we will not publish correspondence sent to us anonymously. We reserve the right to edit letters for purposes of clarity and brevity.

ADVERTISING

ADVERTISING space is available in this publication. Please direct inquiries to the Advertising Manager, People's Computer Company, Box E, Menlo Park, CA 94026. (415) 323-3111

SUBSCRIPTION INFORMATION

U. S. RATES

- \$10/1 yr. (6 issues)
- Retaining subscription @ \$25 (\$15 tax deductible)
- Sustaining subscription @ \$100+ (\$90+ tax deductible)

Please allow 6-9 weeks for your first issue to arrive.

BACK ISSUES

\$2.50 each:
Vol. 6, Nos. 1, 2, 3, 4, 5; Vol 7, Nos. 1, 2
Outside the U.S. add \$.50 per issue.

FOREIGN RATES

- Payments must be in \$US drawn on a US bank.
- \$17 Canada First Class
 - \$23 Rest of World Airmail
 - \$14 World Surface Mail

Delivery of foreign mail is slow and unreliable. We strongly advise airmail.

FOREIGN DISTRIBUTORS OF RECREATIONAL COMPUTING

Vincent Coen, LP Enterprises, 313 Kingston Road, Ilford, 1G1 1PJ, Essex, UK; Rudi Hoess, Electronic Concepts PHY Ltd., 52-58 Clarence St., Sydney, NSW 2000, AUSTRALIA; RS-232 Liz Janik, 186 Queen St., W. Toronto, Ontario M5V 1Z1, CANADA; ASCII Publishing, 305 HI TORIO, 5-6-7 Minami Aoyama, Minato-Ku, Tokyo 107, JAPAN; Ing. W. Hofacker, D815 Holzkirchen, Lindenstr. 8, WEST GERMANY; Jan Nilsson, Hobby Data, S200 12 Malmo, SWEDEN.

Printed by Nowels, Menlo Park, CA

Recreational Computing (ISSN #0164-5846) is published bimonthly by People's Computer Company, 1263 El Camino Real, Box E, Menlo Park, CA 94025. People's Computer Company is a tax-exempt, independent, non-profit corporation, and donations are tax-deductible. Second class postage paid at Menlo Park, California, and additional entry points. Copyright © 1979 by People's Computer Company, Menlo Park, California.

STAFF

EDITORS
Bob Albrecht
Louise Burton
Tracy Deliman
Ramon Zamora
PUBLISHER
Willard J. Holden
ASSISTANT PUBLISHER
Sara Werry
PRODUCTION MANAGER
Carole Cullenbine
ARTISTS
Aleeca Harrison
Ann Miya
Judith Wasserman
TYPESETTERS
Phyllis Adams
Gavin Cullen
Mag Glick
PROOFREADER
Nancy Heubach
CIRCULATION MANAGER
Michael Madaj
WHOLESALE DISTRIBUTION
Robin Allison
SPOT EDITOR
Harry Saal
PROMOTION MANAGER
Betsy Roeth

And a special thanks to all the other folks at People's Computer Company: Delia Daniels, Grant Groberg, Loic Jassy, Ann Merchberger, Nette Chekanasky Wang, Denise Winn.

RETAINING SUBSCRIBERS

Algorithmics, Inc.
David R. Dick
Mark Elgin
John B. Fried
Scott B. Guthery
W. A. Kelley
Brett Wilson

SUSTAINING SUBSCRIBERS

Bill Godbout Electronics
Byte Publications
Paul, Lori and Tom Calhoun

Recreational COMPUTING

Volume 8 Number 2
September - October 1979

formerly
**people's
computers**

Special Features

- 9 **THE SOUNDS FROM TEXAS INSTRUMENTS** by Don Inman
Make your own music on the new TI 99/4
- 16 **ARCHITECTURE OF MULTI-PLAYER GAMES** by Todd Voros
Good fences do good "Universes" make
- 22 **CRYPTARITHMS** by John Davenport Crehore
Who is DADDIDWJM?
- 28 **ROUND 3: PASCAL COOLLY COUNTERS** by David A. Mundie
The great PASCAL/BASIC debate continues
- 32 **SEE WHAT YOU HEAR & HEAR WHAT YOU SEE** by Herb Moore
Dynamic color graphics on the new Atari

Articles

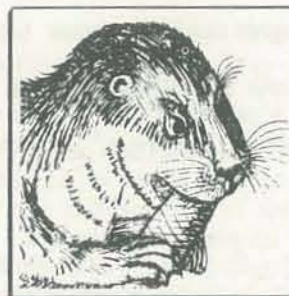
- 24 **AN APPLE PILOT INTERPRETER** by Arthur Wells
A complete listing in Applesoft™ Basic
- 40 **GANDALF** by Ralph Roberts
The Wizard of North Carolina, built by kids
- 48 **THE OUTSIDE CONNECTION** by Doc Plumber
Using the TRS-80 to monitor your home
- 51 **BASIC MASTER COMMAND LIST** by R. T. McLean, A. M. Lopez
J. Schmit, and D. Griffen
Transportable BASIC for PETs, Apples, TRS-80s and SOLs.
- 53 **SPANISH BASIC** by Jim Day
SI 1=5 LUEGO VA A 20
- 64 **APPLE II'S 3 M'S** by Chuck Carpenter
Memory, monitor and machine language: an introduction
- 67 **NEW HAMPSHIRE'S INTREPID HOBBYIST** by Robert Howarth, Jr.
One man's defense of computer games

Games & Stuff

- 31 **TRS-80: GAME OF LIFE** by G. E. Fleming
Colonize your computer
- 38 **NEWETT AWL AND THE GOAT**
PART II: THE SOLUTION by Gordon French
And the green grass grows all around . . .
- 42 **DESIGNING ANIMAL GAMES** by Mike Gabrielson & Tom Martin
Do big birds nest in binary trees?
- 54 **SPOT: THE SOCIETY OF PET OWNERS & TRAINERS** by Harry Saal
Fantasy role playing programs for the PET
- 56 **GRAPHIC TRIPLES FOR APPLE II** by Jim Day
If Pythagoras had a computer . . .
- 57 **APL MASTERMIND** by Nick Cooley
Small programs for giant computers
- 71 **RESPONSES TO THE NEWETT AWL CHALLENGE**
The denouement of the microbus mystery tour

Departments

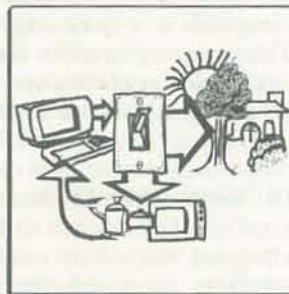
- 4 **EDITOR'S NOTES & LETTERS**
- 56 **FUTUREPLAY**
- 58 **REVIEWS**
- 62 **PROGRAMMER'S TOOLBOX**
- 68 **ANNOUNCEMENTS**



pg. 42



pg. 28



pg. 48



pg. 16

COVER by Aleeca Harrison



Editors' Notes

Extra Pages! Special Features! Changes! News! This issue of *RC* contains all of the above, and more.

The magazine is eight pages larger than normal this time. The expansion was made to accommodate two feature articles on the new Atari and TI home computers, and still leave room for all the regular stuff. If you've been waiting to get a peek at some of the capabilities of these new machines, here is your chance.

The issue also highlights two other articles: "Architecture of Multi-Player Games" and "Cryptarithms." In recent *RC*s, the "Universe" article by Les LaZar and "Cryptarithms" by Jack Crehore, have created a flood of reader response. Todd Voros, co-creator of *FORTRAN Man*, takes the concepts presented in "Universe" to the next level. He proposes an architecture for the implementation of such a system. Jack Crehore, our spry 88-year old author of puzzles, digs again into his bag of mind-benders. Looks like he will become a regular feature in *RC*. If you sent in solutions to the first set of cryptarithms, check Jack's article for your name.

What about changes? Well, the dragon is losing one of its heads. This temporary craziness is prompted by Louise Burton's departure from the masthead of *RC*. Louise is off to do some traveling, expand her freelance activities, and generally have a good time. Good luck, Louise! The other dragons wish you well.

The dragon is sprouting a new head, though, a red one, Tracy Deliman. She is busy working on several books, doing research with a San Francisco organization, freelancing, and starting to learn about computers. Her special interests are in holistic health and medical anthropology. Welcome, Dragoness!

What about the news? The bad news is that *FORTRAN Man* didn't make it into this issue, but will RETURN in the Nov-Dec *RC*. The good news is that we did receive one entry in our "dragon art" contest. (I believe that's good news!) More on the contest in the next issue . . .

Ramon Zamora Tracy Deliman
Louise Burton Bob Albrecht

CORRECTION

In the May-June 1979 issue of *RC*, we mentioned that the Metagame Hunt was available *free* directly from its author, Michael Richter. Alas, that is no longer true! The game is now available from:

Programma International, Inc.
3400 Wilshire Blvd.
Los Angeles, CA 90010
(213) 384-0579

Letters

ARTHUR STRIKES AGAIN

Dear Dragons and Dragonesses:

First, thank you for putting my article in such a pretty setting ("Epic Games on Modest Computers," March-April, 1979).

Second, let me say that while I am in favor of epic games, I don't think they should all be universe-size or populated with hundreds of mythical creations. For many of us, making it through the week without abrasive confrontations or other aggravations is an epic accomplishment. Some of the games should concern real life situations to which players are permitted to try different solutions, using different techniques (e.g., aggression or compromise) or different personalities; fantasies can abound in this area.

Lastly, I note that all of the epic games to date involve war or fighting or conquering or some other form of aggressive competition. Shame, shame. We need games without such attributes. My modest proposal is a game called "Ho Hum." This game is played by any number of people communicating by modem. Players pretend they are at a social gathering. Each player can say whatever he wants whenever he wants. The objective is to be so boorish, trite and boring that no one wants to talk to you any more. Players drop out when they can't stand it any longer or are bored out of their minds. The last player left is the winner. As you can see, no one forces another to capitulate. All action is voluntary.

I am working on two other similar games to be called "Suicide" and "Disease." Believe it or not, some fool has suggested that these programs may not enjoy a large market.

Arthur Wells, Jr.
428 13th St.
Suite 610
Oakland, CA 94612

BASIC DIALECTS BEWILDER

So glad to receive my first copy of *Recreational Computing* (May-June '79) and I enjoyed most of it.

First, I found a BUG on Pg. 15 ("BASIC vs PASCAL vs BASIC"). Line 190 says IF V=2 THEN 670, but there is NO Line 670; my Sorcerer just gives me "? SN ERROR in 190" (Syntax Error). Sorry, Charlie, the good BASIC is not perfect. What is the right one: 270, 370, 470, or 570?

Second, it seems to me nowadays everybody just cares about the software for 3 major micros: Apple, PET and TRS-80, and nobody cares about others such as the magic Sorcerer, which I believe also is a superior micro with all kinds of graphic goodies. Can you do something about this?

Third, every program written in either PET, Apple or TRS-80 BASIC always contains something that only exists in its own BASIC vocabulary. To me, it is a pain just to try to use something that is non-existent in my micro BASIC. Every author always says that it's very easy to transport to another version of BASIC, but God knows how. So, can you suggest any book or literature or whatever to solve this problem? The whole case is like the car door handle which can't be exchanged even though it's from the very same auto company of same year. All these BASICs were developed by Microsoft, and all called "Standard" BASIC (as far as I know), but why do they have to make such a headache for users? Poo-hee!

Fourth, but not the least, I love your magazine. Keep it good!

Tim Huang
19401 East Burnside Hy.
Portland, OR 97233

First, the error was in the original. Looks like a typo for 610(?), but maybe Mr. Mundie is right—BASIC programs can be tough to debug, especially without REM statements. Write to David. His address is with his article in this issue.

Second, with this issue I have been on the magazine for one year. Not one article on the Sorcerer has crossed my desk!

Third, we have an article on a BASIC Master Command list—a subset of BASIC that works on a bunch of machines. If the article is not in this issue, look for it in Nov.-Dec. RC.

Fourth, great!

—RZ

HELP FOR TORNADO VICTIMS

Dear fellow computer enthusiasts:

In the recent tornado which wreaked unholy havoc on our city, many of us in the Wichita Valley TRS-80 Users Group lost our computers, our tape and disk library of software, and our library of computer books and periodicals. Even our club's own library of software and publications was destroyed.

We all have plans to replace our personal computers and software, but at this time I am particularly interested in trying to help our club replace its loss.

Any club, publisher, software producer, or individual who wishes to do so, may contribute non-cash items, such as software, back issues of computer publications, and books on computers.

J. Wesley B. Taylor, Secretary
Wichita Valley TRS-80 Users Group
P.O. Box 4391
Wichita Falls, Texas 76308

IS IT 'THEFT' OR IS IT FREEDOM?

I am writing in response to the letter from Programmer Anonymous #8, Mahwah, NJ (March-April, 1979). Writing programs to simulate a system's logged-off mode is no indication of master programmer status. Such programming activities are nothing more than petty thievery. I am disturbed that your editorial staff failed to take the opportunity to discourage the illegal use of computers.

Joanne R. Hugi
Director, User Services
Computing Center
University of Oregon

Hmmm . . . If enuff people learn how to violate those systems which invade our privacy, perhaps those systems will not happen — power to the people! And, especially, power to kids whose lives might be stomped upon by establishment-centralized data banks. — The Dragon



AUTHORS PROTEST REVIEW

We were shocked to read Harry Saal's review of our new book, *32 BASIC Programs for the PET Computer*, in his monthly SPOT column (May-June, 1979). His criticism probably has misled many readers, and we would like to set the record straight.

Mr. Saal's review focuses on what he calls our "lack of expert programming style." This he explains as a general absence of two things from our programs: explanatory REM statements and instructions for program usage.

These objections are somewhat short-sighted, since they indicate a misunderstanding of the purpose of our book. We are attempting something new—the distribution in book form of BASIC software for a specific microcomputer. As indicated in the book's preface, our major goals were:

1. to provide a broad-based library of useful, creative programs (applications, education, games, graphics, math, etc.);
2. to have them work "as is" on a PET (thus taking advantage of and working within the constraints of PET graphics, screen page formatting, etc.);
3. to provide complete program documentation (including how each user can, if desired, modify the programs to suit his individual needs and tastes).

Some innovative ideas are needed to achieve these goals. Indeed, we thought long and hard about the style that both our book and our programs would have.

We could easily have placed explanatory REM statements and usage instructions within the programs. Instead, we placed complete program documentation in the book, not in the programs. This had some definite advantages. The programs would be shorter. Thus, the user would have less typing to do, and the subsequent loading time from tape would be faster. Program memory space would be conserved,

allowing more extensive program expansion and individual customizing as described in the book. Also, we could squeeze several programs into the 4K PET, a machine still supported by Commodore at the time the programs were being developed. In addition, full documentation in the book would increase the need to buy the book instead of "illegitimately borrowing" the programs on tape from someone else.

In any case, if a user feels he wants explanatory REM statements and usage instruction in the programs, he or she can easily construct them from the appropriate documentation in the book.

Basically, our complaint is that Mr. Saal failed to give our book a responsible review. Our own book reviews for various computing magazines have always striven to give readers two essentials: the purpose of the book, and how well the author achieved his purpose. The first lets the reader know if the book is *potentially* of interest and the second helps him decide if it is *actually* of interest. In reviewing our book, Mr. Saal has ignored the former and has misled the reader about the latter.

Please don't misunderstand our objections. Every reviewer certainly has the right of responsible criticism. If he feels an author's purpose would not benefit many people, fine. If he feels a book is not well executed toward its purpose, fine. But we cannot accept criticism that ignores the intention of our book. It is as if we published a road atlas and were then criticised for not explaining how to drive.

Tom Rugg and Phil Feldman
P.O. Box 24815
Los Angeles, CA 90024

IMPROVING VOICE RESPONSES

Most 'Steemed Dragon,

I was quite interested in Mr. Pollard's article, "Peter Can Now Read," in the May-June issue. I do have a suggestion about the voice response cassette used. I have found that (at least for the Kansas City interface used by my machine) the computer will ignore any voice signals on the data cassette. Thus, if the switch that cuts off the speaker when the ear-phone jack is used is defeated with a "short" wire, both the computer and the student can listen to the tape. This allows the recorder to be stopped when a data block is recognized and means that the data block can also contain the delay parameters and spelling for the next spoken word on the tape. Thus, the word list can be changed without rewriting the program; it also eliminates any problems with the visual and audio words falling out of step.

Errors in making the lesson tape should be rarer, easier to catch and correct. I use this technique myself to provide verbal documentation between programs on a tape, or to provide interesting backgrounds for certain games (maniacal laughter, passages from Bach or Grieg, etc.).

I also wanted to comment on Harry Saal's lack of appreciation for structured design and Jim Day's sadistic fondness for obscure coding (sure it works, but if it didn't, how much longer would it take to figure out why?), but many others have been heard from on that perennial quibble.

Bennett Rutledge
1201 North Pierce Street
Arlington, Virginia 22209

THE PERILS OF PUBLISHING

Yesterday I received my May-June issue of *RC*. I enjoyed most of the items I have read so far, particularly "Hunt" and "Inspector Clew-So."

Today, upon reading my mail, I received a mild shock. In the June '79 issue of *Creative Computing* I found a game program entitled "Inspector Clew-So." Further investigation revealed that the program, the author, and accompanying article were identical to the *RC* piece.

I bring this to your attention, since it reduces the amount of new software I normally find in *RC* and *CC*, and also because you may have been led to believe *RC* would be the only publisher of the game. On the latter point, I cannot draw any conclusions since I do not know under what circumstances you published it.

Anthony Giancola
12600 Northton Ct.
Upper Marlboro, MD 20870

Yes, we were under the impression that RC would be the only publisher of "Clew-So"—or at least the first one. The moral of this story is: if you are a writer submitting an article to two competitive publications simultaneously, you are obligated to let both sets of editors know. If you don't, duplications like the one above are likely to result. With so much good material out there, no one wants to waste space this way!

TOLKIEN DEBATE CONTINUES

I find Lon Ponschock's letter in the July-August issue quite provocative. I agree with much he has to say and share his bias towards the expression of depth and complexity in human motivation. Escapist fantasy can be shallow, although I won't discredit the relief it offers our already complex, troubled world. Sword and sorcery literature also strikes me as vile, and it disturbs me that the filmed version of *Lord of the Rings* seems to overemphasize this aspect.

I admit, however, to some difficulty in distinguishing between Mr. Ponschock's critique of the film and his critique of the books. I get the impression that he has not read the *Trilogy*, and if so, I wish he had limited his remarks to the film.

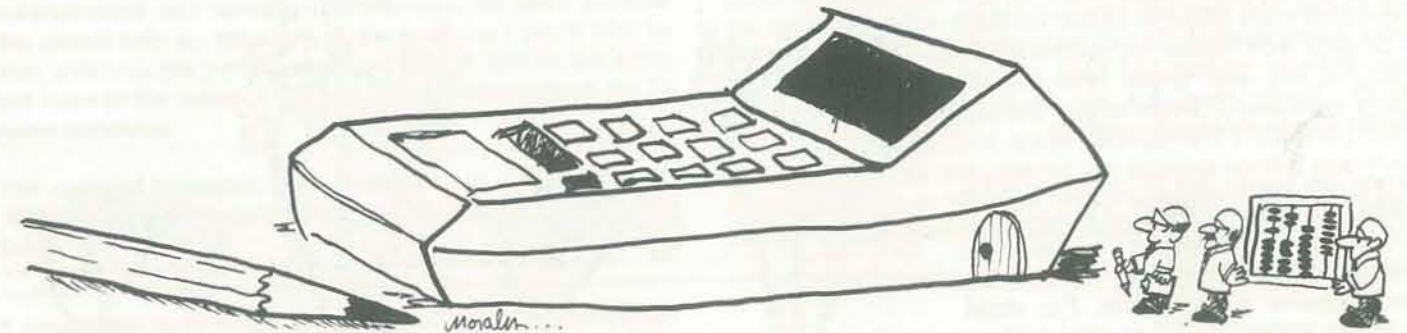
I agree that smugness is usually an irritating trait, in Tolkien fans or anyone else. Tolkien's books are hardly above criticism, and there has been some highly literate criticism of them. Back in the mid-60s when I first read the books (in their pirated Ace Books editions), back when very few knew the difference between an orc and a hobbit-hole, I encountered a different kind of smugness: the smugness of prejudice that declared that if it was fantasy, then it couldn't be good.

I do take strong exception to Mr. Ponschock's "master race" theme, however.

He is flatly wrong here, for neither the film nor the books are operating on this level. We are not "given to believe that the master race is entitled to the power (ring)." The theme is that *no one* is entitled to it, since its corruptive power is absolute. Gandalf's mentor, Saruman, is transformed from a white wizard into a scheming demonic force; Boromir actually betrays the Fellowship while under its lure; poor, pathetic Gollum is a case study in its evil effects. Only the hobbits, simple-minded as they are, are pure enough to carry the ring and then *only* to its destruction. Even then, Frodo is tempted so sorely that the Fellowship is in constant danger from within as well as from without.

There is motivation in the story, but it is not on an ordinary, personal level. It is on a heroic, transpersonal level—common, not to sword and sorcery, but to the genre of medieval verse romance. It is on a mythological, archetypal level. Duality on this level is more than the righteous ignorance of us versus them, hobbits versus orcs, Aryans versus Jews, but rather a metaphor for the life process itself. Our age is now engaged in its own war of the ring. High ideals are not enough, but need to be coupled with courage, selflessness, and wisdom, especially in regard to the corrosive edge of power. I hope our heroes, sung and unsung, emerge as victorious as Frodo the Hobbit.

Michael Madaj
Menlo Park, CA 94025



SIMPLICITY SELLS

Comment on Jim Day's letter in May-June issue: I can read the BASIC statements on page 51 in March-April issue; I cannot read his. Maxim: KISS (Keep it simple, stupid). No offense Jim; you'll learn.

Dave Hawkins
7340 Maringo
Dallas, TX 75227

NEEDED: AUTOMATED BROWSING

Did you ever try to look up a piece of technical information in a library? If it's a small library, they probably don't have it. If it's a large library, chances are pretty good that most of the technical books are kept in closed stacks. This means you have to know the name of the book you want to read before you can read it. So you look in the card catalog under ENCABULATORS, or whatever, and try to guess which of the 75 books listed is the one you want.

The data in the card catalog are so skimpy that all 75 of the books sound almost identical. Oh sure, they list "Introduction to Encabulating," "Popular Encabulators," "Recent Advances in Encabulronics," etc. Now, how do you tell which one lists the annular grillage coefficients for solid-state encabulating profilometers? Maybe five or six out of the 75. But which ones?

If all 75 books were sitting in an open stack, you could flip through the tables of contents and quickly weed out the deadwood. That is, you could if they let you re-shelve the books. Pile up 75 books on a table, and they will kick you out on your dewey decimal. So you pick one or two titles at random and hope you get lucky. Surely there has got to be a better way to access information!

Some day books will be stored on video disks, or the equivalent, and accessed and displayed under computer control. But unless librarians get their act together and organize things better, I'm afraid a new system will be as intractable as

the old. In looking through the literature on library science and information retrieval, I see great effort being expended on fancy indexing schemes, etc., but very little seems to have been done to give the end user quick and convenient access to the contents of an automated library.

If the library of the future is to be of any real use, then someone is going to have to come up with the automated equivalent of "browsing." By that I mean the ability to "flip" very quickly through a selected document, beginning at any point in that document. It should also be possible for a terminal user (who may or may not be physically present at the library) to "tag" a document as pertinent, impertinent, etc., and to instantly extract selected portions for later study.

Many other capabilities would, of course, be useful, but a basic browsing capability is a *sine qua non* as I see it.

Jim Day
17042 Gunther Street
Granada Hills, CA 91344



HELP FLORIDA EDUCATORS!

The Florida Educational Computing Project, which is supported by the State of Florida, has recently approved a project for the evaluation and implementation of a microcomputer based instructional computing system. I am writing to

you in hope that we will be able to contact those who have education-oriented software developed for microcomputers.

We are looking for both CAI-type material and administrative support programs (e.g. film library inventory/control, word processing, statistical analysis, etc.). We do not have the funds to purchase any software at this time and would, therefore, be willing to certify the return or destruction of any program material loaned to us.

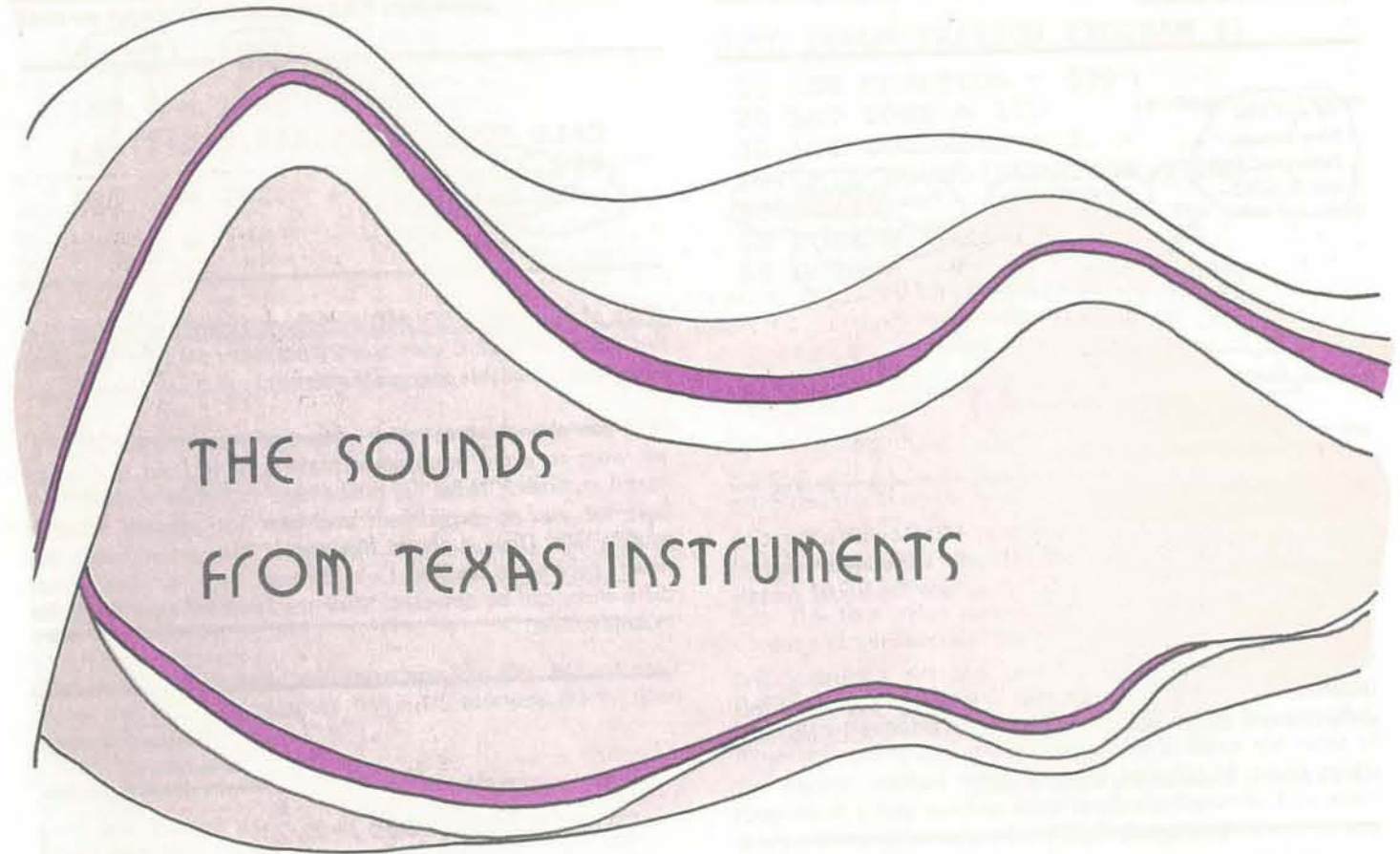
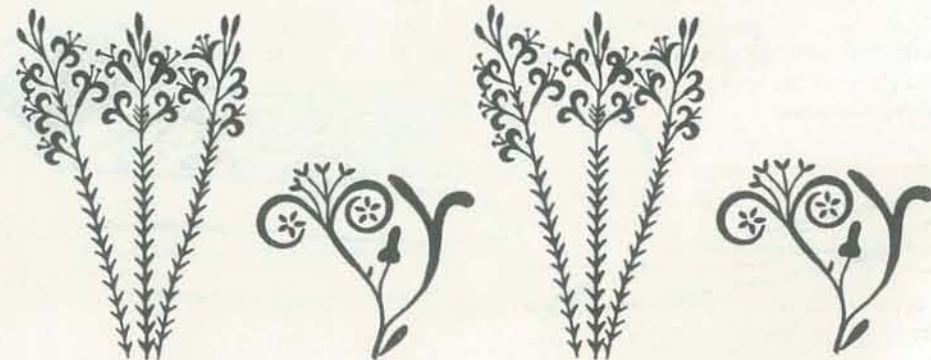
We would prefer programs that are not too dependent on a particular hardware configuration or operating system (if one is required). However, we would like to hear about any programs running on 6502, 6800, or 8080/8085/Z80 machines.

The outcome of this project will be a catalog listing all the acceptable software packages we receive, their evaluation and their source of distribution. This catalog will be made available to all educational institutions in the State of Florida and to any other interested educational systems.

If you have software you wish to submit for evaluation and inclusion in our catalog or if you have questions concerning our project please contact me—

Dr. Nelson J. Towle
Sarasota County Schools
2409 Hatton Street
Sarasota, Florida 33577

Phone: (813) 953-5000 ext. 322



THE SOUNDS FROM TEXAS INSTRUMENTS

Here is a first installment of tutorial information on the Texas Instruments Computer. Later, we will have articles dealing with the color and graphical capabilities of this new machine.

We want to welcome Don back to the pages of RC. There is a rumor (I'm starting it!) that Don may soon appear on the masthead of this magazine . . .
— RZ

BY DON INMAN

Be patient—Texas Instruments is making sound waves with the announcement of their TI-99/4 home computer. The color and sound capabilities of the TI machine promise hours of fun, entertainment and learning opportunities. We can't produce the sounds here on the pages of the magazine—you'll have to wait until you get your hands on a TI-99/4. But we can point out some of the methods that will produce sounds from the TI home computer.

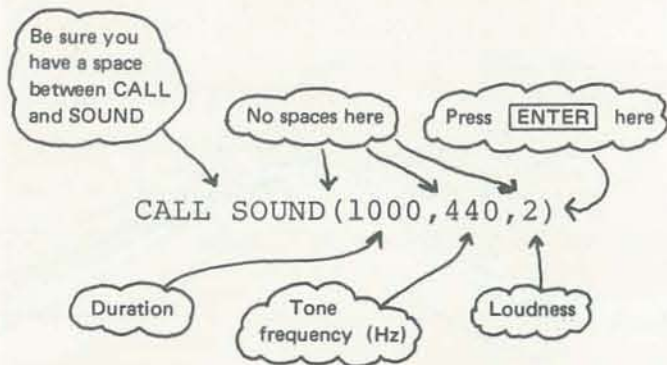
The material presented here is taken from *Introduction to TI BASIC*, a book to be published by Hayden Book Company, Inc.*

* *Introduction to TI BASIC*; Inman, Zamora, Albrecht, and Dymax; Hayden Book Company, Inc.; 50 Essex St.; Rochelle Park, NJ 07662.

Sounds from the computer may be created in either the Immediate Mode or the Program Execution Mode. The sound results from a CALL statement in TI BASIC which uses an external subprogram to execute sound.

Amazing feats of magic seem to result from the use of the two simple words CALL SOUND. With the CALL SOUND statement, you can produce sounds over a range of several octaves, covering frequencies of 110 to more than 44,000 Hertz. Since one Hertz (abbreviated Hz) is equal to one cycle per second, sounds can vary from 110 cycles per second (A below low C on a piano keyboard) to over 44,000 cycles per second (well above human hearing limits). You can also control the duration and the volume of the sound. The duration of the sound ranges from 1 to 4,275 milliseconds. Since one thousand (1,000) milliseconds equals one second, the duration range could be stated as being from 0.001 to 4.275 seconds. Volume selections are scaled from 0 to 30. Zero and one produce the same sound level and are the loudest. Thirty produces the quietest level. (Remember too, that since the TI-99/4 plays through a TV monitor, the volume control of the monitor has ultimate control over the sound level.) Up to three tones and one type of noise may be produced simultaneously over a given time duration.

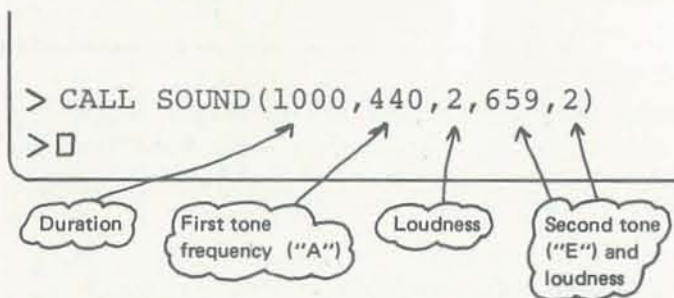
In the Immediate Mode, you need no line numbers. Just type in the statement:



The example above produces a note of 440 Hz with a duration of 1,000 milliseconds (one second) and a loudness value of two (quite loud!). Musicians call this note "A above middle C."

To play two notes at once, you add the frequency and loudness values to those of the first note. See how this enhances the sound (when you get your TI-99/4 computer).

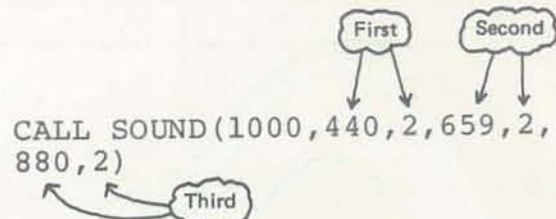
Here's how it looks on the video display.



Note: Because the statement above contains exactly 28 characters (letters, spaces, and symbols), the cursor moves down to the next line as soon as you type the close parenthesis symbol. The TI computer displays 28 text characters per line. Be sure that you remember to press **ENTER**.

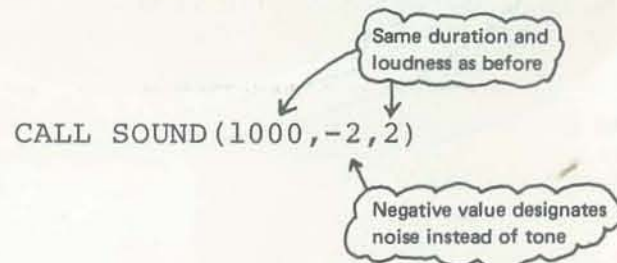
Note that you type the duration parameter (the number code that determines how long the sounds last) only one time—at the beginning of the CALL SOUND statement. Arbitrarily, both of the sounds must last for the same length of time. On the other hand, you can vary the loudness parameters. What would happen if you typed 5, instead of 2, for the second note's loudness. If you have a TI-99/4 and try it, you will find out that the second note will be quieter than the first.

Well, that didn't sound too bad. Let's go one step further and try three notes at once.

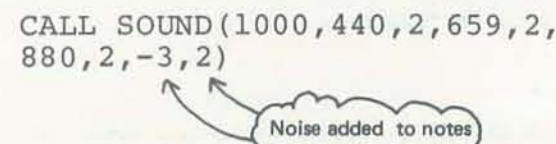


(Part of this CALL SOUND statement extends to the second line, since TI BASIC uses only 28 positions per line. This gives large, clear, readable text on the screen.)

You can also produce noise instead of music notes. Usually, we want to avoid noise when making music, but it may be useful at times. "Noise" is rather hard to define in words. It's best for you to experiment and hear for yourself what it sounds like. Does it sound like static? Play around with the noise parameter awhile (-1 through -8) to hear what differences can be detected. Vary the loudness also. Try these examples first:



You can simultaneously produce up to three tones and one "noise" over a given time duration.

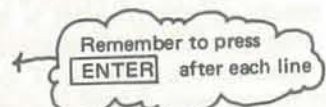


To provide more flexibility, you may use variables, rather than numeric constants, in the CALL SOUND statement. For example, let's use these variables.

- T = time (duration)
- V = volume (loudness)
- C = 262 (middle C on the piano)
- E = 330 (E)
- G = 392 (G)

Then we type in the following LET statements:

```
LET T = 1000
LET V = 1
LET C = 262
LET E = 330
LET G = 392
```

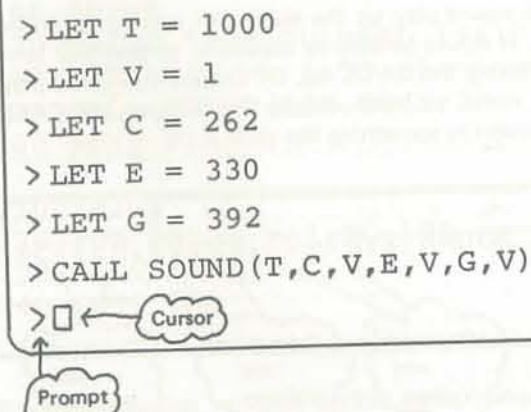


Next comes the CALL SOUND statement.

```
CALL SOUND (T, C, V, E, V, G, V)
```



The video display will look something like this, but we can't show the sound. You must try it. It sounded pretty good through our TV speaker.



By experimenting with other values for duration, tone, volume, and noise within the required range of values for each you can experience a variety of sounds. (A list of musical note frequencies is included at the end of this article.) You'll soon be able to create imaginative sound effects for use in your future programs. The Immediate Mode is helpful for this type of experimentation.

PROGRAMMING

After you have experimented awhile in the Immediate Mode, you'll want to try a few programs. The CALL SOUND statement works the same way in a program as it does in the Immediate Mode. When you're ready to enter a program, type NEW and press **ENTER**. The first task in the program is to assign values to the variables that you'll use.

GOTO DEMONSTRATION PROGRAM #1

```
10 LET DURATION = 500
20 LET TONE = 110
30 LET LOUDNESS = 2
40 CALL SOUND (DURATION, TONE, LOUDNESS)
50 TONE = TONE + 15
60 GOTO 40
```

Go back and make a new sound

Before running the program, you should remember that the program will be running in an endless loop (lines 40-60). It will terminate in one of two ways:

1. You may terminate it at any time by pressing **SHIFT C** on the keyboard.
2. If the tone value goes out of range (above 5500), automatic termination will occur.

If you run the program, listen to the tones that are made. Your ears may object to the sounds created by Program #1. After this you'll appreciate Program #2. Since the notes of the normal musical scale are not exactly 15 units apart, Program #1 may produce some unpleasant sounds. To correct this we'll try other values for TONE that provide a one-octave scale.

GOTO DEMONSTRATION PROGRAM #2

```
10 LET T = 500
20 LET V = 2
30 C = 262
40 D = 294
50 E = 330
60 F = 349
70 G = 392
80 A = 440
90 B = 494
100 HIC = 523
```

```
200 CALL SOUND (T, C, V)
300 CALL SOUND (T, D, V)
400 CALL SOUND (T, E, V)
500 CALL SOUND (T, F, V)
600 CALL SOUND (T, G, V)
700 CALL SOUND (T, A, V)
800 CALL SOUND (T, B, V)
900 CALL SOUND (T, HIC, V)
```

```
950 GOTO 200
```

Again, this is an endless loop. Press **SHIFT C** to stop it.

Run the program. Up, up, up you go until you reach high C. Then the GOTO statement at line 950 sends you back to middle C to start over. STOP THE PROGRAM! Then reverse the order of lines 200-900. If you run the program with those lines reversed, the notes go down, down, down from high C to middle C. Once again, line 950 causes the sounds to be repeated over and over again. Monotonous, but much easier than practicing the scales on the piano.

You have played an octave both up and down. Now let's put them together so that you go Up, Down, Up, Down, etc.

We replace lines 200-950 with the following.

```

200 CALL SOUND(T,C,L)
205 CALL SOUND(T,D,L)
210 CALL SOUND(T,E,L)
215 CALL SOUND(T,F,L)
220 CALL SOUND(T,G,L)
225 CALL SOUND(T,A,L)
230 CALL SOUND(T,B,L)
235 CALL SOUND(T,HIC,L)
240 CALL SOUND(T,B,L)
245 CALL SOUND(T,A,L)
250 CALL SOUND(T,G,L)
255 CALL SOUND(T,F,L)
260 CALL SOUND(T,E,L)
265 CALL SOUND(T,D,L)
270 GOTO 200

```

UP, UP, UP
DOWN, DOWN, DOWN
Go back and repeat it all

Your fingers naturally object to all the repetitive typing. So we'll have to find a shorter, more powerful way to accomplish the same result. A logical way to do this would be through a FOR-NEXT loop with READ and DATA statements. The program becomes much shorter.

MODIFIED SCALE PROGRAM

```

10 LET T = 500
20 LET V = 2
30 FOR X = 1 TO 15
40 READ N
50 CALL SOUND(T,N,V)
60 NEXT X
70 END

```

N for note

```

100 DATA 262,294,330,349,392
110 DATA 440,494,523,494,440
120 DATA 392,349,330,294,262

```

Up and Down in order

To make the scale play continuously up and down you could:

1. Change line 70 to: 70 RESTORE
2. Add line 80: 80 GOTO 30
3. Change line 120 to: 120 DATA 392, 349, 330, 294
4. Change line 30 to: 30 FOR X = 1 TO 14

The beauty of this program is that you can now play any tune that you want by changing the DATA list and the upper limit for the FOR-NEXT loop. You could also custom design the program to fit your needs by reading in changing durations and volumes and by including the variables T and V in the FOR-NEXT loop and DATA statement.

CUSTOM FITTED NOTES PROGRAM

```

10 FOR X = 1 TO 8
20 READ T,N,V
30 CALL SOUND(T,N,V)
40 NEXT X
50 RESTORE
60 GOTO 10

```

Duration, Note and Volume in order

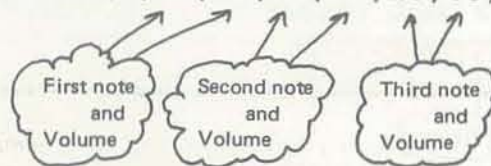
```

100 DATA 100,262,2,100,294,3
110 DATA 200,330,2,100,349,3
120 DATA 100,392,2,100,440,3
130 DATA 100,494,2,500,523,2

```

This program would play up the scale with varying durations and volumes. It would be easy to modify it to play any tune desired by altering the DATA list. Of course, you could play simultaneous notes to make chords by changing the CALL SOUND statement to something like this:

```
30 CALL SOUND(T,N1,V1,N2,V2,N3,V3)
```



Your DATA statement would then have to include values for each note and volumes for each chord.

GAME TIME

Let's take a slight diversion and play a game. Surely, you have played Guess My Number or one of its many variations. A novel version of the number guessing game can be created on the TI-99/4 using its sound capabilities. The following program plays a tone between 131 cycles per second and 247 cycles per second. Your role is to guess the frequency of the tone. The program lets you know if your guess is lower, higher or equal to the frequency of a random tone that is generated by the computer. When you guess the frequency of the note correctly, the program plays the note three times and begins the game again with a new random note.

RANDOM NOTE GUESSING GAME

```

10 CALL CLEAR
20 TONE = INT(117*RND)+131
30 PRINT "OK, I HAVE A TONE"
40 PRINT
50 PRINT "THE TONE IS--"
60 CALL SOUND(100,TONE,2)
70 INPUT "WHAT IS YOUR GUESS": GUESS
80 IF GUESS = TONE THEN 160
90 IF GUESS > TONE THEN 125
95 CALL SOUND(100,GUESS,2)
100 PRINT "TOO LOW!!"
110 PRINT "TRY A HIGHER TONE"
120 GOTO 40
125 CALL SOUND(100,GUESS,2)
130 PRINT "TOO HIGH!!"
140 PRINT "TRY A LOWER TONE"
150 GOTO 40
160 PRINT
170 PRINT "YOU GUESSED IT!!"
180 FOR PLAY = 1 TO 3
190 CALL SOUND(100,TONE,2)
200 NEXT PLAY
210 PRINT
220 FOR DELAY = 1 TO 500
230 NEXT DELAY
240 GOTO 10

```

For an interesting variation of this game, remove lines 100, 110, 130 and 140. These lines contain visual messages telling you if you are high or low. Without the messages, you have to play the game by ear.

The tone limits can be changed in line 20. Also, you may want to add the RANDOMIZE statement to create a new series of random tones each time that you run the program. If so, just enter this line:

```
15 RANDOMIZE
```

No provision was made in the program to detect invalid inputs (line 70). IF-THEN statements can be used to detect and reject invalid inputs.

RANDOM MUSIC

You can make the computer play some interesting (but not necessarily enjoyable) "music" by letting it choose random notes.

RANDOM NOTES PROGRAM

```

10 CALL CLEAR
15 LET C = 262
20 LET D = 294
25 LET E = 330
30 LET F = 349
35 LET G = 392
40 LET A = 440
45 LET B = 494
47 LET C2 = 523
50 RANDOMIZE
55 NOTE = INT(8*RND)+1
60 TIME = INT(1000*RND)+100
65 VOLUME = 2
70 IF NOTE = 1 THEN 200
75 IF NOTE = 2 THEN 300
80 IF NOTE = 3 THEN 400
85 IF NOTE = 4 THEN 500
90 IF NOTE = 5 THEN 600
95 IF NOTE = 6 THEN 700
100 IF NOTE = 7 THEN 800
105 NOTE = C2
115 CALL SOUND(TIME,NOTE,VOLUME)
120 GOTO 55
200 NOTE = C
210 GOTO 115
300 NOTE = D
310 GOTO 115
400 NOTE = E
410 GOTO 115
500 NOTE = F
510 GOTO 115
600 NOTE = G
610 GOTO 115
700 NOTE = A
710 GOTO 115
800 NOTE = B
810 GOTO 115

```

Run the random notes program (as long as you can tolerate it). When you're ready to stop press **SHIFT** **C**.

Now that you've let the computer play its own music, why not play some music of your own? Let's see if we can modify the Random Notes Program to provide a Musical Interlude Program where you can choose your own notes.

MUSICAL INTERLUDE PROGRAM

```

10 CALL CLEAR
15 C = 262
20 D = 294
25 E = 330
30 F = 349
35 G = 392
40 A = 440
45 B = 494
47 C2 = 523

50 INPUT "NOTE ": A$
    ← leave a space

70 IF A$ = "C" THEN 200
75 IF A$ = "D" THEN 300
80 IF A$ = "E" THEN 400
85 IF A$ = "F" THEN 500
90 IF A$ = "G" THEN 600
95 IF A$ = "A" THEN 700
100 IF A$ = "B" THEN 800
105 IF A$ = "C2" THEN 900
    ← Compare input to note

110 GOTO 50 ← Go back, incorrect input

115 CALL SOUND(100,NOTE,2) ← Play the note
    
```

```

117 FOR DELAY = 1 TO 50
119 NEXT DELAY
120 GOTO 50

200 NOTE = C ← Set NOTE equal to key pressed.
210 GOTO 115

300 NOTE = D
310 GOTO 115

400 NOTE = E
410 GOTO 115

500 NOTE = F
510 GOTO 115

600 NOTE = G
610 GOTO 115

700 NOTE = A
710 GOTO 115

800 NOTE = B
810 GOTO 115

900 NOTE = C2
910 GOTO 115
    
```

If you run the Musical Interlude Program, the computer will ask you for a note. You then type in the note desired (A,B,C,D,E,F,G or C2) followed by the **ENTER** key.

For example, when the screen shows:

NOTE

and you press **A** **ENTER**, the note A will play. The screen keeps a record of the keys that you depress:

```

NOTE C
NOTE D
NOTE E
NOTE F
etc.
    
```

You can then make a nice, clean method to STOP THE MUSIC by using an IF-THEN-ELSE statement at line 110.

```

Change → 110 IF A$ = "S" THEN 950 ELS
        E 50
        .
        .
        .
        .
        .
        .
Add → 950 END
    
```

Now if no "legal" note has been pressed, the computer checks A\$ to see if you want to stop. If you have typed an S, the computer goes to line 950 and stops. If you have not typed a legal note or an S, there was an error in your input. The computer then returns to line 50 for a new input.

Additional applications of the sound features of the TI-99/4 can be found in the Hayden book: *An Introduction to TI BASIC* referenced on the first page of this article.

TABLE OF SOUND FREQUENCIES

Note	Frequency cycles/sec	Note	Frequency cycles/sec
A	110	A	880
A#	117	A#	932
B	123	B	988
C	131	C	1047
C#	139	C#	1109
D	147	D	1245
E	165	E	1319
F	175	F	1397
F#	185	F#	1480
G	196	G	1568
G#	208	G#	1661
A	220	A	1760
A#	233	A#	1865
B	247	B	1976
C	262	C	2093
C#	277	C#	2218
D	294	D	2349
D#	311	D#	2489
E	330	E	2637
F	349	F	2794
F#	370	F#	2960
G	392	G	3136
G#	415	G#	3322
A	440	A	3520
B	494	B	3951
C	523	C	4186
C#	554	C#	4435
D	587	D	4699
D#	622	D#	4978
E	659	E	5274
F	699	F	5588
F#	740		
G	784		
G#	831		



- Take your favorite character—or let the computer create one for you!
- Let the Book of Lore guide you through a DUNJONQUEST™ within the Temple.
- Decide to fight the monsters or grab the treasure and run—but don't think too long—they'll come after you!

The Vault of the Dead is but one of the many dark and fearsome mysteries within the ruined Temple of Apshai. The Temple of Apshai is your first adventure in the DUNJONQUEST™ series of fantasy role playing games. DUNJONQUEST™ is a complete game system and The Temple of Apshai is a complete fantasy adventure game for you and your microcomputer.

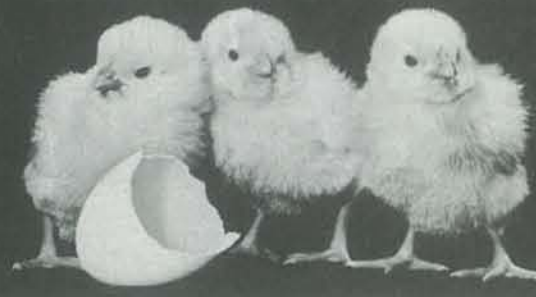
OVER 200 ROOMS!
OVER 30 MONSTERS!
OVER 70 TREASURES!

The Temple of Apshai—for the TRS-80 (Level II, 16K) and PET (32K) microcomputers.

Ask your local dealer or send a check for \$24.95 to:

Automated Simulations-Department Q,
P.O. Box 4232, Mountain View, CA 94040
California residents please add 6% sales tax.

We've hatched a nestful of new tiny-c products.



Now you can really expand your horizons with the tiny-c structured programming language. The tiny-c owner's manual (including 8080 and PDP-11 source code and tiny-c in C) is still just \$40. And we've added these new formats to really egg you on: TRS-80 Level II SYSTEM Format Cassette, CP/M Diskettes with 8080 Source, PDP-11 Diskette, North Star 5" Diskette, KIM and SYM cassettes. And there's more, plus lots to come. Order your tiny-c owner's manual today and get the whole story. Call or write: tiny c associates, P.O. Box 269, Holmdel, N.J. 07733. (201) 671-2296. You'll quickly discover tiny-c is all it's cracked up to be.

tiny
C

New Jersey residents include 5% sales tax. Visa and MasterCard accepted. Include charge plate number with order.

Architecture of Multi Player Games

BY TODD VOROS

The "Universe" article, by Les LaZar, (RC, March-April 1979) generated considerable response and correspondence. Todd sends us this "short letter to the Editors" outlining the next steps in developing a Universe game context. The information in this article is thought-provoking and well developed. We know that there is a universe of "Universe" fans somewhere out there. Read this and send us your contributions, thoughts, opinions, and suggestions for what in the world to do with the "Universe."
— RZ

I read your article on "Universe" with considerable interest. Since I am currently involved in the implementation and maintenance of complex multi-user teleprocessing systems, I would like to contribute some technical suggestions for the implementation of multi-player "Universe" games. The games may or may not include remote teleprocessing "Universe" links.

From experience, one of the most troublesome problems for the user of system services is the burden of specification. By this, I mean the machine-specific environmental knowledge that is necessary to support the intended applications. The information that is usually needed is often not part of the "working" section of a program concerned directly with implementing the user code. Generally, the more system services available the more specification is required. Specification becomes the price tag for implementation flexibility.

This situation rapidly becomes intolerable in a small machine environment. One cannot force an implementor to learn the idioms of every small machine, or restrict a user to a wide set of flexible services on a single machine.

This was the problem faced by the original implementors of high-level languages. Standardization was the solution. Handle machine-specific differences invisibly for the casual user, at the expense of some additional overhead, in terms of memory and/or execution speed.

Consider the following: The price of memory is steadily declining as semiconductor fabrication technology is able to mass-produce denser memory systems. As quantity increases, price must drop. This bodes well for the home hobbyist. Thus, when designing systems, memory must no longer be a prime concern—ease of user interface to the system must be at the forefront of design concepts! This must be true for both the implementor (the user of system services) and the end user (the user of implementor services).

With this in mind, I offer the following design suggestions:

The concept of FENCING	(To limit complexity)
Communication MAPPING	(To simplify connections)
Standardized Communication Services	(To minimize implementation efforts.)

Each of these concepts will be discussed in detail. The idea here is that the final architecture should be easily integrable into a high-level language available on micro computer systems. Note that once the architecture is finalized, it's external appearance is similar from machine to machine. However, the machine-specific implementation of internals may be quite different. This shall be our design goal. The specification of the architecture requires a machine-independent language. Portions of the architecture in this letter are annotated in Sketchcode where appropriate. (For information on Sketchcode see PC, Vol. 6, No. 6, May-June, 1978.)

FENCING

This component of the architecture provides a "boundary" to the application's knowledge of its environment. Everything relevant to a particular application is "inside" the fence; everything else—other programs, devices, users, etc. are "outside" the fence and not of concern to the application.

For the multi-player games with inter-user communication, we define the "fence" as the standardized communication specification interface.

Any communication occurring by other applications outside of the "fence" is of no concern to the currently executing application. As a matter of fact, as far as the application is concerned, such communication does not exist—it has no knowledge of anything outside the "fence."

This implies that application-to-application connections cannot occur unless they each define an appropriate "gate" in their fences. Establishment of this "gate" shall be done by invoking the *FENCE MANAGEMENT MANAGER*.

Flexibility is the keyword in evolving a successful microprocessor communication architecture while maintaining standardization. To achieve this, I suggest that we follow an implementation strategy.

First, every gate in the fence is to have a name. These names are known as *resources*.

Second, the *user* specifies the resource name. When a resource is specified, the user supplies the name of a machine-specific subroutine that will *manage* the named resource. *Parameters* shall be passed, at the time the fence "gate" is established, that define machine-specific hardware associated with the resource.

Note that this allows for a very flexible mechanism of integrating new devices or inter-communication mechanisms to be implemented. Generally, the resource manager named is probably a machine-language subroutine to handle some device.

Third, a *resource name* is associated with a resource manager, it remains associated until respecified.

Also note that establishing a resource name ("gate" in the fence) with a resource manager *does not* transfer control to the manager code—it merely logs the resource name and location of the manager code for later usage.

The following is the environmental specification necessary to establish connections. The actual code in the fence management manager itself is rather brief. For example, to establish "terminal "gate" we might code:

```
CALL FENCE ( User-resource-name ,
             keyboard port =    ,
             printer port =    ,
             I/O driver = MYTTYCODE)
```

where MYTTYCODE is the name of the teletype handler in my system. Assume we wish to build a simple intercom between two terminals. The keyboard and printers on each terminal have their own port numbers. Assume that this communication function is to be run by a *single* application.

Then the following might apply:

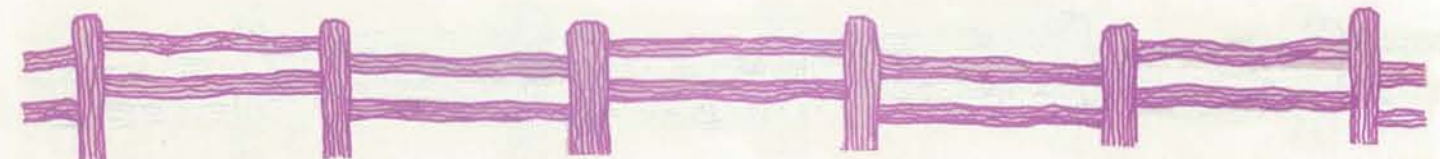
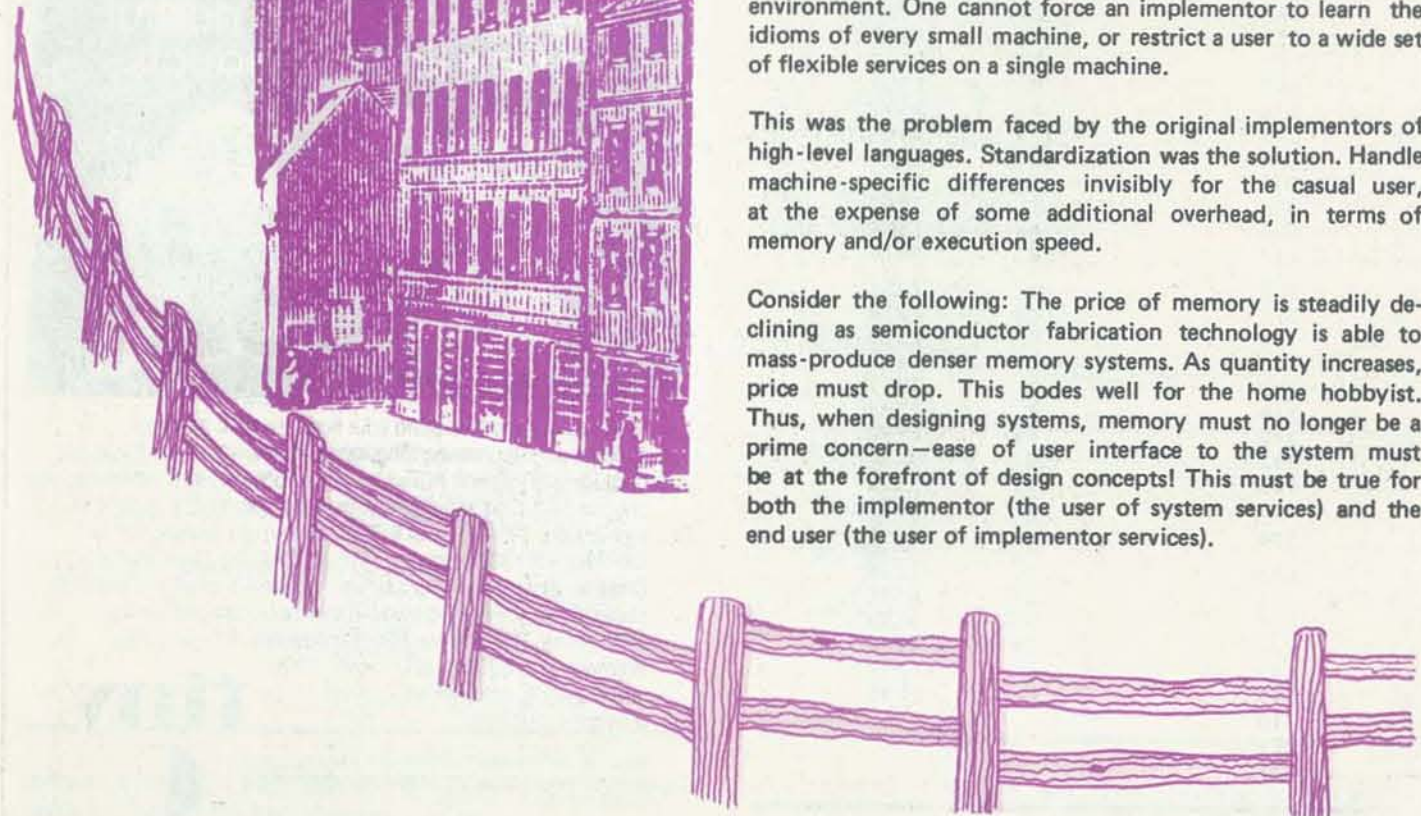
```
Terminal 1
keyboard = port 1
printer = port 2
Terminal 2
keyboard = port 17
printer = port 18
```

We will give Terminal 1 the resource name 'CRT001'
We will give Terminal 2 the resource name 'TTY001'

Then we would need to invoke the Fence Manager twice:

```
CALL FENCE ( 'CRT001' , 1 , 2 , TERMCODE)
CALL FENCE ( 'TTY001' , 17 , 18 , TERMCODE)
```

The architecture proposed does not specify the syntax of the FENCE; the above is merely a hypothetical example of what a machine-specific implementation of that function may look like. The syntax should be adapted based on the machine and high-level language currently being used.



The Sketchcode for the Fencing Manager follows:

```

FENCE MANAGER:  ACQUIRE PASSED PARAMETERS

IF RESOURCE-NAMES TABLE FULL,
  THEN DO;
    ISSUE OUT-OF-ROOM MSG
    TERMINATE APPLICATION

DO WHILE (EMPTY SLOT IN TABLE NOT FOUND)
  EXAMINE ENTRY IN RESOURCE NAMES TABLE
END DO;

STORE USER SPECIFIED RESOURCE NAME INTO TABLE ENTRY
STORE USER SPECIFIED PARAMETERS INTO TABLE ENTRY

IF RESOURCE-MANAGER CODE IS NOT LOCATED,
  THEN DO;
    ISSUE INVALID-FENCE-SPECIFICATION MSG
    TERMINATE APPLICATION

STORE ADDRESS OF RESOURCE MANAGER CODE INTO SAME ENTRY

RETURN;
  
```

One final note on establishing resource-names. The resource *does not* need to be an I/O device! This architecture allows and promotes overlays to be considered as resources. For example, the establishment of an overlay for use by a given application might involve the following machine-specific parameters:

The 'name' of the overlay; the FILENAME on disk of the overlay, and the device on which the overlay resides:

```

CALL FENCE (
  'OVERLAY12' , - resource name
  'MATHPKG.OBJECT', - file name
  FLOPPYDISK2, - device file located on
  DISKRTN) - disk driver handler
  
```

In summary, *FENCING* limits complexity of inter-user and inter-application communication by establishing a common method of specifying, to an application, what resources it can communicate with. *FENCING* does not make actual connections, but specifies the ability to do so. *FENCING* always associates a support subroutine with a resource name. *FENCING* provides parameters for the support subroutine (known as a "resource manager" in this architecture) to perform its job—be that disk I/O; loading overlays; or communicating with terminals or modems. Connections are established via Communication MAPPING—the next topic.

COMMUNICATIONS MAPPING

Communication *MAPPING* is a system-provided service that simplifies actual resource-to-resource communication. An objective of this architecture is to minimize the programmer's effort.

Two new routines, LINK and FREE are suggested. Their implementation is more complex than that of the FENCING MANAGER. The objective of the LINK/FREE manager is to permit the programmer to perform I/O just as he always has done in the past—with READ, WRITE, REWIND, etc. statements (or their equivalent in the language in which he is working) without changes!!

This concept makes this proposed architecture very attractive from the implementation viewpoint of the system user. He does not need to learn strange new I/O methods to talk to another CPU, overlay, or resource. This magic is accomplished by the LINK/FREE manager and here is how it works.

When the user generates his program, and wishes to communicate with a resource, he issues a call to the LINK routine. He then proceeds to issue normal high-level language I/O statements to a 'device' by the LINK manager. This is accomplished by having the LINK manager modify generated high-level language machine code.* Obviously, the author of the LINK manager must have a good knowledge of the internals of the

high-level language. But again, this need only be done *once* for many users of the language and system to benefit from the architecture of this communications scheme. The code is modified to transfer control to the routine specified in the call to the FENCING MANAGER—the *user's resource manager* routine whenever an I/O operation is to be executed.

Example:

```

CALL LINK ('resource name') becomes:
READ X,Y,Z-----CALL user-resource-mgr (X,Y,Z)
WRITE Y,Z-----CALL user-resource-mgr (X,Y,Z)
REWIND
X=X+1
WRITE X
  
```

--- 1st call: With a flag this is a READ.
--- 2nd call: Without flag this is a WRITE.

Note that all I/O requests will access the resource specified in the call to the LINK Manager, in the above example. If I/O is to be performed to another resource, a call to the FREE manager must be made, and another call to the LINK manager must be made specifying the new resource. Actually, LINK and FREE share much common code, so they will probably be packaged together as one routine.

If multiple processes (jobs or tasks) are supported by the system in use it is possible that some process may already have LINKED to a resource desired by another process. In this case, the requestor is delayed until the "current" owner of the re-

* Thus, standardized communications services are provided through an already familiar medium—READ statements, WRITE statements, etc.—that the user has been coding since he learned the language!!

source issues a FREE for that resource. This case is why FREE is a required part of the architecture, and why two LINKS in a row cannot be issued.

The LINK/FREE manager must be capable of (1) locating I/O code generated by the high level language, (2) locating a resource in the Resource Table, (3) dynamically inserting a CALL to the associated user-support subroutine for the resource, (4) determining the type of I/O operation being performed and passing it to the user-resource manager subroutine, and, (5) putting the caller of the LINK manager into a wait, if the resource requested is currently owned by another process.

If an attempt is made to free a resource that has not been linked, a warning is issued by the FREE manager, and the request treated as a 'NO OPERATION' request, and program execution continues.

If an attempt is made to LINK to a non-existent resource, a message is issued and the program terminated, to allow the user to debug his problem. Termination at this point is logical because it is impossible to establish communication with a non-existent resource.

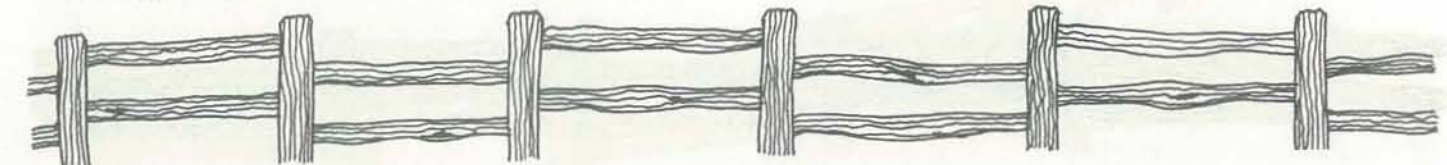
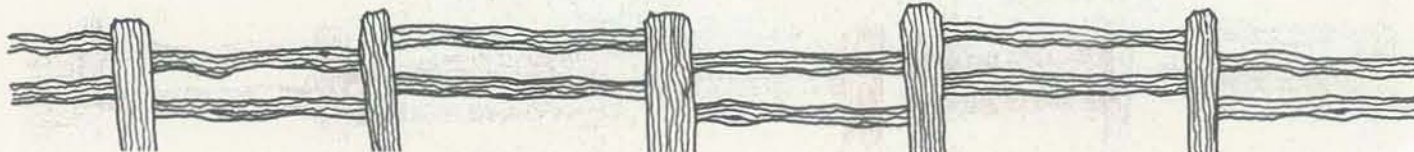
A Sketchcode representation of the LINK/FREE MANAGER follows. It assumes that compiler-generated code is not re-entrant, and that the machine code is in modifiable machine memory. Generally, for small scale microprocessor systems with high-level languages this is a reasonable assumption. To support compilers that generate re-entrant code would require significant extensions to the LINK/FREE manager logic.

```

LINK:  Call VALIDATE ROUTINE(CALLER=LINK)
        LINKING ADDRESS = User's resource manager subroutine
                           associated with specified 'resource'
DO WHILE (END OF PROCESS NOT FOUND),
  EXAMINE INSTRUCTION SEQUENCE
  IF SEQUENCE IS AN I/O CALL,
    THEN DO,
      REMEMBER ORIGINAL I/O SEQUENCE IN A SAVE AREA
      REMEMBER LOCATION OF I/O SEQUENCE IN A SAVE AREA
      REPLACE I/O SEQUENCE WITH CALL TO LINKING ADDRESS
      THAT WILL PASS ORIGINAL SEQUENCE PARAMETERS.
    END DO;
  RETURN

FREE:  Call VALIDATE ROUTINE(CALLER=FREE)
DO WHILE (END OF REMEMBERED ADDRESSES FOR PROCESS NOT FOUND),
  REPLACE ORIGINAL INSTRUCTION AT EACH 'REMEMBERED' ADDRESS
END DO;
RETURN
  
```

Please recall *no* I/O may be performed without the user calling LINK first. Once LINK has been called, all I/O is performed to the specified resource until a call to FREE is made.



Here is the Sketchcode for the VALIDATE subroutine:

```
VALIDATE: LOCATED = 'NO'
          DO WHILE (All resource table entries not checked),
            IF resource-table-entry resource name matches
              caller's resource name,
              THEN DO,
                LOCATED = 'YES'
            END DO,
            IF LOCATED = 'YES',
              THEN DO,
                IF 'FREE' called VALIDATE,
                  THEN DO,
                    INDICATE THIS RESOURCE IS UNASSIGNED
                  ELSE,
                    IF RESOURCE IS ASSIGNED,
                      THEN DO,
                        PUT CALLING PROCESS INTO A 'WAIT' FOR RESOURCE
                        DISMISS TO MONITOR
                      ELSE,
                **Resource available:      SAVE PROCESS IDENTIFICATION INTO RESOURCE TABLE ENTRY
                (asynchronous             INDICATE RESOURCE IS ASSIGNED
                entry by monitor)         RETRIEVE ADDRESS OF USER'S RESOURCE MGR SUBROUTINE
              ELSE,
                IF CALLER = LINK,
                  THEN DO,
                    ISSUE ERROR MSG
                    DUMP APPROPRIATE PARTS OF MEMORY FOR DIAGNOSIS
                    ABORT EXECUTION
              RETURN
```

Note: The sketchcode assumes the existence of a monitor functioning within the machine to provide multiple-process concurrent execution. Thus, several independent processes executing simultaneously could wish to use a specific resource at the same time. The first request will be honored, all other requestors for the resource will be placed into a 'wait' for the resource until the first process releases ownership by issuing a 'FREE' call.

Thus, the sketchcode assumes the monitor also monitors the resources name table and 'wakes up' processes awaiting the use of a waited-for resource. Thus, the VALIDATE sketchcode has an asynchronous (special, monitor only) entry

point which is used by the monitor to continue the execution of a process that had been 'put to sleep' waiting for a resource. This is a minor violation of sketchcode standards. (Shame! Shame! — ED.)

Obviously, the monitor must maintain a queue of processes awaiting use of a resource, the length of the queue being n-1 entries for the n-processes competing for the same resource. First-In-First-Out control of a resource is given to requesting processes.

If the *same* process requests the *same* resource twice, it will go to sleep 'forever'. This is a *deadly embrace*, since the process can never acquire the resource.

CLOSING COMMENTARY

The design of communication systems, machine to machine, or even between processes within the same machine, should, by now, be shown to be more difficult than one might initially think.

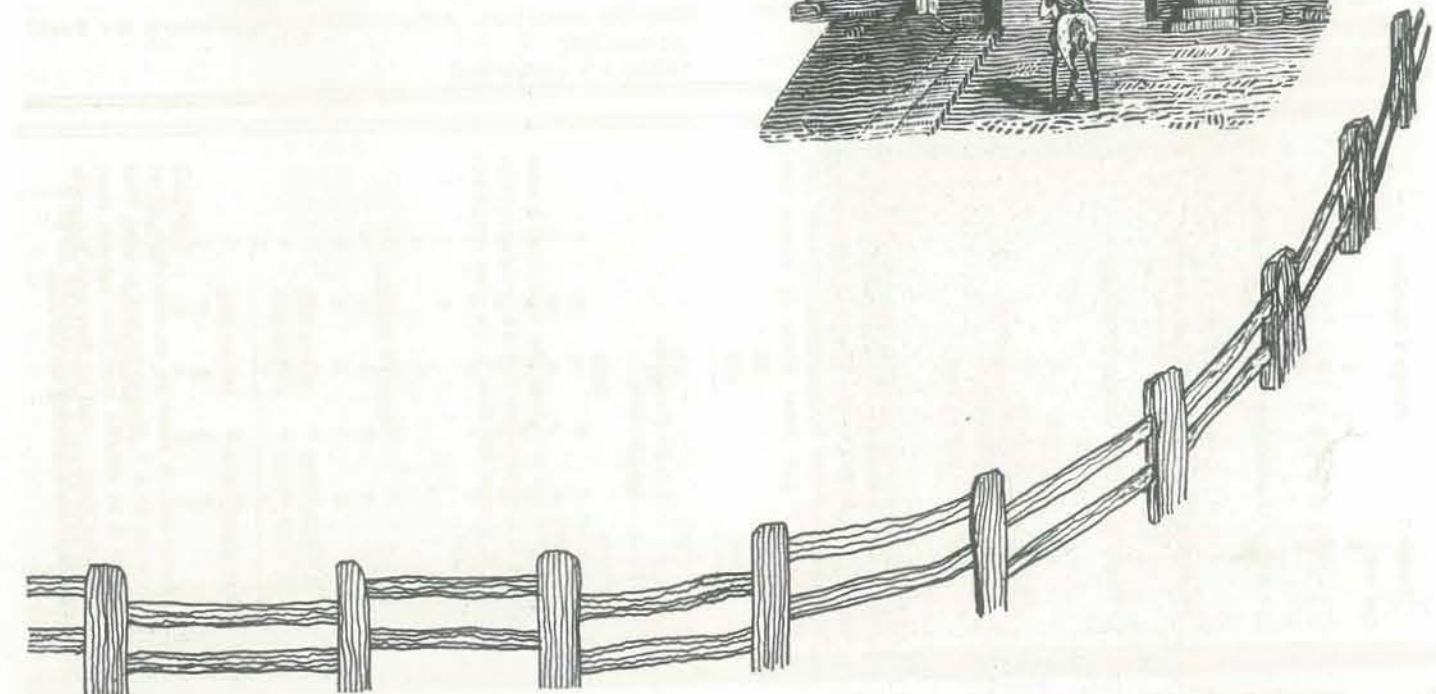
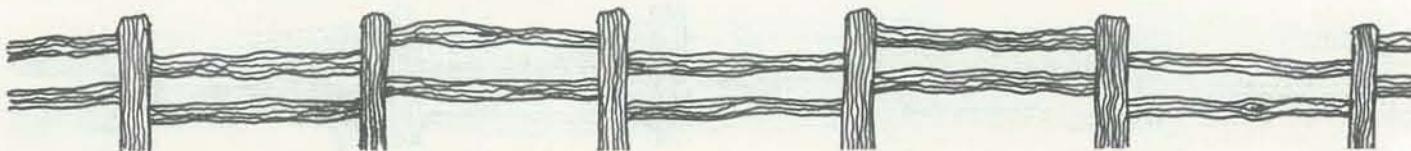
Much effort has been expended by many programmers to solve problems like these. This particular architecture may not satisfy all needs or requirements of microprocessor users or the designers of 'super game networks'. I would be most interested to hear of any suggestions, improvements or implementations of this proposed architecture.

Since you have read this article up to this point, you are obviously either:

- a) An insatiable reader of computer literature.
- b) A computer buff.
- c) A person with too many beers.
- d) All of the above.
- e) Other.
- f) Other than other.

Please circle the correct choice and... It is with the hope that work along these lines may one day bring the Don Quixote Starship to reality. May I suggest a motto appropriate to the closing of this article:

*Computers should work-Not people!!
(And help them play games)*



Cryptarithms

BY JOHN DAVENPORT CREHORE

Looks like Jack's cryptarithms are here to stay! The reader response to the first set of puzzles was outstanding. We even have a computer program writing to us. (SPOCK in the Puzzler's Postcard section). Here are some new puzzles, samples

and examples, the Postcard section, and the solution records of our new puzzlers. We will print solutions to all puzzles for 1979 in the Jan-Feb 1980 issue of RC. Let's hear from all of you! -RZ

NEW PUZZLES

Puzzle 9 (Novice)

$$\begin{array}{r} \text{SS} \\ + \text{SS} \\ \hline \text{MSL} \end{array}$$

Hint: Look at the Samples and Examples.

Puzzle 10 (Adepts)

$$\begin{aligned} M^2 &= \text{FM} \\ M^3 &= \text{DCM} \\ M^4 &= \text{CDBM} \\ M^5 &= \text{RRRM} \end{aligned}$$

Hint: The same five digits uniquely solve all four equations. M^2 has an M in the units position of the answer; only two possible single digit products can do this!

Puzzle 11 (Geniuses)

$$\begin{array}{r} \text{C}^T \cdot \text{ND} \cdot \text{A}^A \cdot \text{T}^U \cdot \text{CB} \cdot \text{D} \cdot \text{TU} \cdot \text{R}^T \cdot \text{ND} \\ \hline \text{U}^T \cdot \text{R}^Y \cdot \text{BE} \cdot \text{TR} \cdot \text{AC} \cdot \text{N}^T \cdot \text{E}^T \cdot \text{AT} \cdot \text{A}^B \end{array} = \text{D}$$

Hint: No hints to geniuses! They compete on equal terms with computers!

Puzzle 12 (Computers)

$$\begin{array}{r} \text{L SUL TUL} \\ \times \text{E SSL TTY} \\ \hline \text{BM MHH HHH} \end{array}$$

Hint: No hints here, either! How many solutions are there?

SAMPLES AND EXAMPLES

Puzzle 1

$$\begin{array}{r} \text{B} \\ \text{B} \\ \text{B} \\ \hline \text{CB} \end{array}$$

Puzzle 1 Solution

$$\begin{array}{r} 5 \\ 5 \\ 5 \\ \hline 15 \end{array}$$

The number 5 is the only digit that works.

Puzzle 2

$$\begin{array}{r} \text{R} \\ \text{R} \\ \text{T} \\ \hline \text{TR} \end{array}$$

Puzzle 2 Solution

$$\begin{array}{r} 9 \\ 9 \\ 1 \\ \hline 19 \end{array}$$

The number 9 is the only digit that works.

Puzzle 3

$$\begin{array}{r} \text{AC} \\ \times \text{AC} \\ \hline \text{FH} \\ \text{JF} \\ \hline \text{YBCH} \end{array}$$

Puzzle 3 Solution

$$\begin{array}{r} 32 \\ \times 32 \\ \hline 64 \\ 96 \\ \hline 1024 \end{array}$$

The letter A can only have the value 3. If A were more than 3, three letters would be needed in place of JF; if A were less than 3, the total would not reach 1,000.

Puzzle 4

$$\begin{array}{r} \text{A, ABB, FCB} \\ + \text{H, KCF, FKB} \\ \hline \text{HM, MAM, MFA} \end{array}$$

Puzzle 4 Solution

$$\begin{array}{r} 8,844,534 \\ + 1,235,524 \\ \hline 10,080,058 \end{array}$$

In column one, A must be an even digit. Why? The H must be a 1. Right? So HM is easy. That solves for B in column one; then column four; then column three... Get it? Stay with it a while and the logic starts jumping off the page.

PUZZLER'S POSTCARDS

I must express my disappointment with the puzzles you gave in issue No. 40... the letters used to represent digits don't spell words. I haven't tried the multiplication puzzles yet. I run on a PET, occupy only 2K of memory, and solved puzzle 5 in 30 seconds. Puzzle 7 took me 40 minutes. I expect to rewrite part of myself in machine language and increase my speed. I'm less than one day old now. One of my favorite puzzles is:

$$\begin{array}{r} \text{SEND} \\ \text{MORE} \\ \hline \text{MONEY} \end{array}$$

SPOCK
c/o CALTECH 130-33
Pasadena, CA 91125

I am curious as to the future of cryptarithms and the correctness of the solutions I am submitting. Please set your tally under my newly adopted puzzler's name: "The Worry Wart" or TWW for short. (I thought it was better than Jewish Mother!)

Marvin Kessler (TWW)
586 Elvis Dr.
San Jose, CA 95123

These are some of my favorite puzzles. I did all these by hand (paper and pencil). I am interested in seeing anyone's computer methods, especially any that don't try all combinations (10 factorial?).

C.A. Moore
1265 Kuehne
Ann Arbor, MI 48103

I have enjoyed solving the few cryptarithmic puzzles I have come across in my twenty-three years. I was pleased to see some in your last issue. It was disappointing that the puzzles were easier than I expected them to be. Puzzle 8 was a bit easier to solve than puzzle 7; perhaps puzzle 8 should be in the adept category. Here is a puzzle, though not difficult, that I like:

$$\begin{array}{r} \text{EH} \\ \times \text{GLK} \\ \hline \text{TTT} \\ \text{RLU} \\ \hline \text{RUG} \\ \hline \text{TPTPT} \end{array}$$

David Hubbard
346 Jean St.
Mill Valley, CA 94941

I enjoyed the puzzles and would enjoy seeing this as a regular column in RC (if restricted to one or two pages). Perhaps there could be a regular puzzle column in RC, not restricted to cryptarithms. Comments on the puzzles:

- The hints on #6 were unnecessary, especially (d).
- It took me about two hours to solve all eight. In retrospect, I did several the hard way first.
- Number 3 has a particularly elegant solution that I noticed when I examined it a second time. It goes like this:

Since Y is a carry, it must equal 1;
Since J has a carry it must be 9 and B must be 0;
Therefore, A = 3 and C ≤ 3 which implies C = 2;
Immediately, F = 6 and H = 4.

Very nice, eh?

My puzzler name is a bit of egotism on my part. It is an English school slang word meaning expert or adept. I found it while browsing through the O.E. D. one day.

Eryk Vershen (Dab)
Menlo Park, CA 94025

PUZZLER'S SOLUTION TABLE

Name	Current Puzzle	Total Solutions
DAB	5	4
TWW	6	4
SPOCK	7	2
D. Hubbard	8	4
C.A. Moore	X	4
S. Schram	X	4
D. Marquis	X	4
B. Baum	X	4
R.H.(?)	X	1
M. Richter	X	1
C.I. Goldman	X	4
K88DU	X	4
Crazy Man	X	4
S.R. McEnter	X	4
DADDIDWJM	X	4

Legend: (●) By Hand
(X) By Computer

Late Notes: The latest batch of letters also contained programs, puzzle solution schemes, and good yuks! The last person (DADDIDWJM) in the table sent a cryptarithm whose solution is his/her name.

$$\begin{array}{r} \text{MAJOR} \\ \times \text{WOMAN} \\ \hline \text{INAWCM} \\ \text{MMCCON} \\ \text{ON IIC} \\ \text{ICJIAO} \\ \text{ACIRI} \\ \hline \text{DADDIDWJM} \end{array}$$

The letters, in numerical order, spell the name!

0 1 2 3 4 5 6 7 8 9

Oh, well!

-RZ

An Apple PILOT Interpreter

BY ARTHUR WELLS, JR.

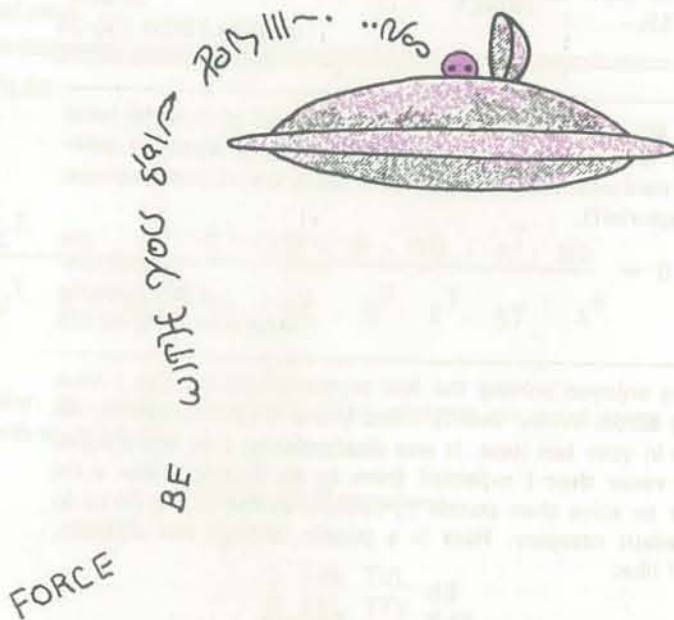
How about PILOT for your Apple written in BASIC? "Easy to do," says Arthur, and proves it by sending along the following program.

Now, it would be delightful if a few of you people would write some programs based on this interpreter. Write something interesting (way out, absurd, whatever...) and send it on to us.

Arthur says if you have questions, call him at (415) 848-4058 or drop a line to 1171 Cragmont Ave., Berkeley, CA 94708. Call or write him even if you don't have any questions. I believe Arthur just likes to talk with lots of people. -RZ

This documentation tells you how to write, use and store programs written in the PILOT language using a PILOT Language Interpreter program on your Apple II.

When the interpreter is run, it asks you for the name of the program that you want. If you want to use an existing program, type in its name and hit the return key. If you misname or mistype the program name, the interpreter will tell you and repeat the request. After the program loads from disk there is a short delay while the program is prepared for execution. The machine then asks "REQUEST?". You can

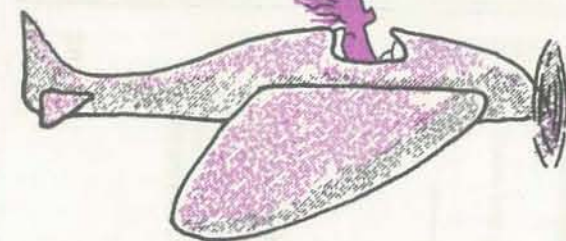


then LIST, EDIT, or RUN the PILOT program. After the program has been run once, if you want to do it again, type REP. Do not type RUN. RUN is used only after loading and editing. REP is used to substantially speed things up. (The reason is explained later.)

If you do not want to run a program but instead want to start writing one, push the return key when the machine asks for a program name. You will be told to start writing your program and a "?" will prompt you. The "?" is asking for a line number. Line numbers from 1 to 255 are acceptable. Although programs cannot exceed 255 lines, this still allows for large useful programs. For instance, the working parts of the interpreter itself consist of less than 255 statements. The "?" prompt at the beginning of a line will accept only a number; if you type something else, it will go back and wait for a number.

After you type the line number hit the return key. The cursor will move over and a "?" will appear. Now type in the command and statement; then press return; enter next line of the program; and so on.

When you are through writing the program, enter the next line number and instead of a statement, type RUN. Hopefully, the program will work.

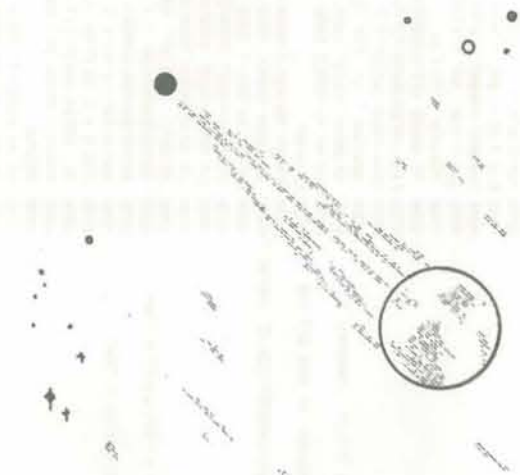


PILOT LANGUAGE ELEMENTS

- T/ For *Text* or *Type*. Prints whatever appears after "/". Commas and colons *cannot* be used because of limitations of Applesoft (the interpreter language). T/ statements can be 255 characters long.
- A/ For *Ask*. Requests user input. Input may be assigned to a variable and printed later, using a variable name. A variable is a character @ plus any combination of alphanumeric characters. For example, the instruction "A/ @ NAME" assigns the user input to the variable "@ NAME".
- M For *Match*. Matches a string with the last user input. More than one match word or phrase is permissible; each must be followed by a semicolon. Example: M/NO; NOPE; I DON'T KNOW; This is a "moving window" match, which means that if the match word is a part of the user input, a match will still be made. For example, the match word RAIN will show a match with the following user inputs: BRAIN, DRAIN, RAIN, RAINING, STRAIN. The match word EVOL matches with EVOLUTION and REVOLUTION.
- ME/ For *Match Exactly*. Matches a string with the user input, but requires that match be exact. The match word RAIN will only show a match with the input RAIN.
- J/ For *Jump*. Program execution jump to elsewhere, specified by * and the name of the new location. Example: J/* BELOW jumps to wherever * BELOW is located in the program.
- R/ For *Return*. Returns execution of program to line after the Jump that was last executed. J/ and R/ together are like GOSUB, and RETURN in BASIC.
- E/ For *Exit*. Stops program execution.
- C/ For *Count*. Used with two variables I and J. Either variable can be set to zero by C/ZI or C/ZJ. Either can be increased or decreased by using the format C/I + number or C/I - number, where number is any real number. Example: C/J - 12 subtracts 12 from the value of J. The current value of I or J can be printed by using # I or # J in a T/command. Example: If I = 6 and @ PLAYER = Arthur, then "T/@PLAYER has #I POINTS" will print "ARTHUR HAS 6 POINTS".
- CK/ For *Check*. Checks to see if the value of I or J is = > or < a specified value. Example: CK/I=5 checks to see if I=5.
- TY, TN
AY, AN
JY, JN
MY, MN
- If match or check occurs, commands with Y (for yes) execute; if no match or check occurs commands with N (for no) execute.

PILOT CONTROL COMMANDS

- LIST Asks "which lines" then lists them. If a program has been run, commands will not be listed, just statements.
- EDIT Will ask "which lines" then display part of program requested and show a "?" for line number and then statement. To edit a line, retype what you want. To delete a line, type line number and then leave command space blank (just hit the return key). You can insert new lines between other lines.
- RUN Used to run a program *only* after LOAD or EDIT.
- REP Used for repetitions of program after it has been run.
- NEW Clears out the old program and accepts a new one or allows user to start writing a program.
- STOP Returns control to Applesoft.
- SAVE Saves program to disk.



PILOT HINTS & SUGGESTIONS

- A. If you are in the EDIT mode, you can LIST by using that command after a line number. You can also get out of the EDIT mode by typing REQ after a line number. This will get you back to "REQUEST".
- B. Put subroutines at the end of your program for faster execution. The interpreter looks for jump labels from the end of your program up.
- C. After a variable you can only have a space, a period, or an apostrophe.
- D. When a RUN is typed, the interpreter separates your program into two arrays, one for commands (T/, A/, etc.), the other for the statements, and then commences to execute the program one line at a time. Setting up the arrays after loading a program or editing takes a few seconds.
- Once the arrays are set, the program can be rerun quickly. That is why REP is used. However, an EDIT command reassembles the commands and statements. Thus a RUN is necessary to set up the arrays after an EDIT.

Listing

```

5  CALL - 936
10  VTRB 5: HTRB 11: PRINT "PILOT INTERPRETER"
20  PRINT: HTRB 11: PRINT "COPYRIGHT 1978 BY"
30  HTRB 12: PRINT "ARTHUR WELLS, JR."
33  PRINT
45  PRINT : PRINT
50  PRINT "THIS PROGRAM PERMITS YOU TO WRITE AND RUN PROGRAMS IN THE 'PILOT' LANGUAGE."
55  PRINT "LUCID AND ENTERTAINING DOCUMENTATION ACCOMPANIES THIS PROGRAM. IF YOU GOT THE
PROGRAM WITHOUT THE DOC-"
70  PRINT "MENTATION PLEASE LET ME KNOW AND I HILLIETHER SEND YOU THE DOCUMENTATION OR SUEYOU"
75  PRINT
80  PRINT "MY ADDRESS IS 428 - 13TH ST., OAKLAND, CA 94612
90  FOR I = 1 TO 3000: NEXT I
200  A = 0: Z = 0: Z1 = 1: C1 = 0: C2 = 0: H = 0: E = 0: E1 = 0: D = 0
220  C1(1) = "E": C1(2) = "R": C1(3) = "T": C1(4) = "M": C1(5) = "J": C1(6) = "N": C1(7) = "I":
230  C1(14) = "R": C1(15) = "C": C1(16) = "K": C1(17) = "W"
240  C1(8) = "V": C1(9) = "N": C1(10) = "M": C1(11) = "W": C1(12) = "R": C1(13) = "M"
241  DIM D1(25), V1(25), R1(25), S1(25), F1(30), T1(25)
242  DIM P1(25), O1(25), I1(12), V1(25), Z1(12), R1(7)
244  O1 = "": REM CONTROL D
246  PRINT D1: "NONCONC. 1.0"
285  CALL - 936: VTRB 2
287  ONERR GOTO 1860
288  IF FIN < 100 THEN FIN = 100
289  FOR X = 1 TO FIN: P1(X) = "": NEXT X: CALL - 936
290  PRINT "WHICH PROGRAM DO YOU WANT?": INPUT FILE#
295  IF FILE# < > "" THEN 3000
297  ONERR GOTO 1860
300  CALL - 936: VTRB 2
305  HTRB 7: PRINT "START WRITING YOUR PROGRAM"
306  PRINT
308  ONERR GOTO 1800
310  CV = PEEK (37)
313  INPUT A
316  VTRB: CV + 1: HTRB 5
330  INPUT P1(A)
340  IF P1(A) = "RUN" THEN 405
342  IF P1(A) = "END" THEN 768
345  IF P1(A) = "LIST" THEN 752
359  CV = CV + 1
400  GOTO 310
405  ONERR GOTO 1850
410  CL = 0: AL = A - 1
415  PRINT "I'M GETTING IT READY."
450  FOR A = 1 TO AL
455  IF LEFT$(P1(A), 2) = "" THEN O1(A) = "": GOTO 540
460  FOR B = 1 TO 3
470  IF MID$(P1(A), B, 1) < > "+" AND MID$(P1(A), B, 1) < > "*" THEN 430
480  GOTO 540
490  IF MID$(P1(A), B, 1) < > "/" THEN 520
500  O1(A) = LEFT$(P1(A), B - 1): P1(A) = MID$(P1(A), B + 1)
510  GOTO 540
520  NEXT B
540  NEXT A
549  CALL - 936
555  VTRB 3
590  FOR A = 1 TO AL
595  IF O1(A) = "" THEN 630
600  FOR C2 = 1 TO 20
610  IF O1(A) = C1(C2) THEN 640

```

```

630  NEXT C2
640  IF C2 = 1 THEN 695
650  ON C2 GOSUB 210-1340: 940, 1170, 1490, 1460, 1480, 910, 930, 1140, 1160, 1300, 1320, 1580, 4000, 4500, 1730
690  NEXT A
695  FIN = AL + 1
696  PRINT: PRINT
698  GOTO 1000
700  FOR H = 1 TO FIN
710  IF O1(A) = "" THEN 740
720  P1(A) = O1(A) + " " + P1(A)
730  O1(A) = ""
740  NEXT A
745  IF R1 = "EDIT" THEN 792
750  IF R1 = "SAVE" THEN 2000
759  CL = 0
760  INPUT "REQUEST?": R1#
762  IF R1 = "SAVE" THEN 700
765  IF R1 = "EDIT" THEN CALL - 936: GOTO 740
770  IF R1 = "NEW" THEN 287
775  IF R1 = "RUN" THEN CALL - 936: GOTO 405
780  IF R1 = "DEF" THEN CALL - 936: GOTO 590
790  IF R1 < > "LIST" THEN 860
791  ONERR GOTO 752
792  PRINT "WHICH LINES?":
793  INPUT S: F
795  FOR A = 5 TO F
800  IF LEFT$(P1(A), 1) = "" THEN 850
825  HTRB 2: PRINT R1: HTRB 6: PRINT P1(A)
830  NEXT A
855  IF R1 = "EDIT" THEN 308
860  IF R1 = "STOP" THEN END
870  GOTO 760
910  IF M = 1 THEN 940
920  GOTO 1100
930  IF M = 1 THEN 1100
940  FOR Z = 1 TO LEN (P1(A))
950  IF MID$(P1(A), Z, 1) = CHR$(64) THEN GOTO 1067
955  IF MID$(P1(A), Z, 1) = "R" THEN 1065
960  NEXT Z
970  GOTO 1090
980  FOR Z1 = 2 TO 40
990  IF MID$(P1(A), Z1, 1) = CHR$(32) THEN Z1 = Z1 - 1: GOTO 1020
992  IF MID$(P1(A), Z1, 1) = "" THEN Z1 = Z1 - 1: GOTO 1020
994  IF MID$(P1(A), Z1, 1) = " " THEN Z1 = Z1 - 1: GOTO 1020
996  IF MID$(P1(A), Z1, 1) = "*" THEN Z1 = Z1 - 1: GOTO 1020
1000  NEXT Z1
1010  GOTO 1090
1020  VE = MID$(P1(A), Z1, Z - Z + 1)
1030  FOR H = C TO 1: STEP - 1
1040  IF VE < > V1(H) THEN 1061
1050  PRINT I1(H)
1051  IF MID$(P1(A), Z1 + 2, 1) = "R" THEN PRINT " ",
1052  Z1 = Z1 + 1
1060  GOTO 960
1061  NEXT H
1065  PRINT MID$(P1(A), Z1, Z - Z1)
1066  Z1 = Z + 1
1068  IF MID$(P1(A), Z1, 1) = "*" THEN 1072
1070  PRINT I1(A): Z1 = Z + 2: GOTO 960
1072  PRINT I1(A): Z1 = Z + 2: GOTO 960
1087  PRINT MID$(P1(A), Z1, Z - Z1): Z1 = Z1 + 1: GOTO 980
1090  PRINT MID$(P1(A), Z1)
1094  Z1 = 1
1100  RETURN
1140  IF M = 1 THEN 1170
1150  GOTO 1260
1160  IF M = 1 THEN 1260
1170  M = 0: E1 = 1

```

```

1180  FOR E = 1 TO 40
1190  IF MID$(P1(A), E, 1) = " " THEN 1210
1200  GOTO 1230
1210  IF MID$(P1(A), E, 1) = E1 + 1 = Z# THEN 1250
1220  E1 = E + 1
1230  IF MID$(P1(A), E, 1) = "" THEN 1260
1240  NEXT E
1250  M = 1
1260  RETURN
1300  IF M = 1 THEN 1340
1310  GOTO 1420
1320  IF M = 1 THEN 1420
1340  FOR Z = 1 TO 10
1350  IF MID$(P1(A), Z, 1) < > "0" THEN 1400
1360  C = C + 1: V1(C) = P1(A)
1370  INPUT I1(C)
1380  Z# = I1(C)
1390  GOTO 1420
1400  NEXT Z
1410  INPUT Z#
1420  RETURN
1460  IF M = 1 THEN 1490
1470  GOTO 1530
1480  IF M = 1 THEN 1530
1490  FOR D = 1 TO AL
1500  IF O1(D) < > "" THEN 1540
1510  IF LEFT$(P1(A), D) < > LEFT$(P1(A), D) THEN 1540
1515  RET = R
1520  A = D
1530  RETURN
1530  RETURN
1540  NEXT D
1550  PRINT "YOU JUMPED TO H LABEL THAT DOESN'T EXIST."
1560  GOTO 700
1570  END
1580  H = RET: RETURN
1600  VTRB: CV + 1: GOTO 310
1720  M = 0: E1 = 1
1725  FOR E = 1 TO 40
1730  IF MID$(P1(A), E, 1) = " " THEN 1745
1740  RETURN
1745  L = E - E1
1750  FOR X = 1 TO LEN (Z#) - (L - 1)
1755  IF MID$(Z#, X, L) = MID$(P1(A), E, L) THEN M = 1: RETURN
1760  NEXT X
1765  E1 = E + 1
1770  GOTO 1725
1775  RETURN
1800  VTRB: CV + 1: GOTO 310
1850  PRINT: PRINT "A MISTAKE HAS BEEN MADE."
1851  GOTO 760
1860  PRINT: PRINT "THERE IS NO PROGRAM BY THAT NAME": PRINT: GOTO 250
2000  PRINT "I'M WORKING..."
2001  FOR A = 1 TO AL: H1(A) = STR$(A)
2005  NEXT A
2010  FOR A = 1 TO AL
2015  IF LEFT$(P1(A), 2) = "" THEN 2050
2050  T1(A) = R1(A) + " " + P1(A)
2060  NEXT A
2100  PRINT "WHAT ARE YOU GOING TO CALL THIS PROGRAM?": INPUT FILE#
2105  T1(A1 + 1) = "2222"
2110  PRINT D1: "OPEN": FILE#
2120  FOR A = 1 TO AL + 1
2130  IF T1(A) = "" THEN 2160
2140  PRINT D1: "WRITE": FILE#
2150  PRINT T1(A)
2160  NEXT A
2199  PRINT D1: "CLOSE": FILE#
2199  GOTO 760

```

```

3000  PRINT D1: "OPEN": FILE#
3010  CTR = 0
3020  FOR A = 1 TO 255
3030  PRINT D1: "READ": FILE#
3040  INPUT S1(A)
3050  IF S1(A) = "2222" THEN 3090
3055  IF S1(A) = "" THEN 3080
3067  CTR = CTR + 1
3080  NEXT A
3090  PRINT D1: "CLOSE": FILE#
3100  FOR A = 1 TO CTR
3105  IF S1(A) = "" THEN 3250
3110  FOR Z = 1 TO 4
3120  IF MID$(S1(A), Z, 1) = "R" THEN 2000
3130  NEXT Z
3200  H1(A) = LEFT$(S1(A), Z - 1)
3210  P1 (V1 (H1(A))) = MID$(S1(A), Z + 1)
3250  NEXT A
3592  A = VAL (R1(CTR))
3595  A = A + 1
3650  GOTO 760
3700  GOTO 410
4000  FOR Z = 1 TO LEN (P1(A))
4010  IF MID$(P1(A), Z, 1) = "2" THEN 4050
4020  IF MID$(P1(A), Z, 1) = "1" THEN 4100
4030  IF MID$(P1(A), Z, 1) = "*" THEN 4300
4040  NEXT Z
4050  FOR Z = 2 TO LEN (P1(A))
4055  IF MID$(P1(A), Z, 1) = "*" THEN 4000
4060  NEXT Z
4070  I = 0: I1# = STR$(I): RETURN
4080  J = 0: J1# = STR$(J): RETURN
4100  T = VAL (MID$(P1(A), 2, LEN (P1(A))))
4110  I = I + 1
4120  I1# = STR$(I)
4130  RETURN
4300  T = VAL (MID$(P1(A), 2, LEN (P1(A))))
4310  J = J + 1
4320  J1# = STR$(J)
4350  RETURN
4500  FOR Z = 1 TO LEN (P1(A))
4510  IF MID$(P1(A), Z, 1) = "*" THEN 4800: NEXT Z
4520  FOR Z = 2 TO LEN (P1(A))
4540  IF MID$(P1(A), Z, 1) = "*" THEN 4560
4545  IF MID$(P1(A), Z, 1) = "C" THEN 4580
4550  IF MID$(P1(A), Z, 1) = "D" THEN 4570
4555  NEXT Z
4560  T = Z: V = VAL (MID$(P1(A), T + 1)): GOTO 4600
4565  T = Z: V = VAL (MID$(P1(A), T + 1)): GOTO 4650
4570  T = Z: V = VAL (MID$(P1(A), T + 1)): GOTO 4680
4600  M = 0: IF V = 1 THEN M = 1
4610  RETURN
4650  M = 0: IF V > 1 THEN M = 1
4660  M = 0: IF V < 1 THEN M = 1
4685  RETURN
4800  FOR Z = 2 TO LEN (P1(A))
4810  IF MID$(P1(A), Z, 1) = "*" THEN 4900
4820  IF MID$(P1(A), Z, 1) = "C" THEN 4930
4830  IF MID$(P1(A), Z, 1) = "D" THEN 4950
4940  NEXT Z
4900  T = Z: V = VAL (MID$(P1(A), T + 1)): GOTO 4960
4930  T = Z: V = VAL (MID$(P1(A), T + 1)): GOTO 4970
4950  T = Z: V = VAL (MID$(P1(A), T + 1)): GOTO 4980
4960  M = 0: IF V = J THEN M = 1
4965  RETURN
4970  M = 0: IF V > J THEN M = 1
4975  RETURN
4980  M = 0: IF V < J THEN M = 1
4985  RETURN

```

Round 3: PASCAL

Coolly Counters

BY DAVID A. MUNDIE

Round 1 of this discussion began in People's Computers, Jan-Feb 1978 with David's original article. Round 2 was in the May-June 1979 issue of Recreational Computing. David returns this issue with Round 3.

Along the way there have been many preliminary skirmishes and feigned battles. David addresses some of those in the article also. I can't wait for Round 4. It gets better each time!
— RZ



My primary interest in promoting PASCAL was to hasten the day when I would have a PASCAL machine in my kitchen. So, the day I placed my order for a PASCAL Microengine I vowed to withdraw from the ongoing debate over PASCAL's merits compared to those of BASIC. However, the "BASIC backlash" in your pages has prompted me to break that vow, and to offer the following observations.

THE DRAGON IS DROPPED

Mr. Albrecht's comment in the RC March-April issue is little more than a potshot. He claims that the following BASIC program segment, cited by PASCAL supporter Joe Felsenstein, is an "unreal" example:

```
100 IF X = 3 THEN 500
200 LET Y = Y - 1
300 LET Z = Z - 1
400 GO TO 700
500 LET Y = Y + 1
600 LET Z = Z + 1
700 REM
```

Mr. Albrecht asks rhetorically, "Who would write a BASIC program like that?" The answer, of course, is: "Lots of people, including Mr. Albrecht himself!" See for example, the program on page 111 of his self-teaching guide, *BASIC* (second edition, John Wiley and Sons, 1978).

Mr. Albrecht suggests the following alternative code:

```
100 IF X = 3 THEN Y = Y + 1: Z = Z + 1
110 IF X <> 3 THEN Y = Y - 1: Z = Z - 1
```

It is hard to tell exactly what Mr. Albrecht's point is. If it is simply that his proposed alternative is shorter, then we must give him the point. However, if he is suggesting that BASIC's IF-statement is as good as or better than PASCAL's, then I must disagree:

- One must assume that programmers will often need to have several lines of code whose execution is dependent on the value of a boolean expression. What would Mr. Albrecht recommend then? Twenty-five lines of code beginning "IF X = 3 THEN" followed by twenty-five more beginning "IF X <> 3 THEN"? PASCAL simply brackets these statements with "begin" and "end" and is done with it.

- The program fragment proposed by Mr. Albrecht will be interpreted differently by different versions of BASIC. Some versions will treat "IF X = 3 THEN Y = Y + 1" as a complete statement, and proceed to execute "Z = Z + 1" no matter what the value of X is. Would Mr. Albrecht suggest the following code for users of such BASICs?

```
100 IF X = 3 THEN Y = Y + 1: IF X = 3 THEN Z = Z + 1
110 IF X <> 3 THEN Y = Y - 1: IF X <> 3 THEN Z = Z - 1
```

Is this elegant programming? This problem of ambiguous IF-statements is just one of many which confront those trying to spruce up a line-oriented language with constructs which are essentially statement-oriented. Whatever minor standardization problems PASCAL may have, at least its control structures are clearly defined.

- Many versions of BASIC will not accept Mr. Albrecht's program segment at all—namely, those which do not allow multiple statements on one line. In short, what is the *only* version of this program fragment which will execute in all BASICs with the desired result? *The original version given by Mr. Felsenstein.* Anyone writing BASIC programs for general use would be well advised to adopt it. I contend that it is not an "unreal" example at all.

While we are on the subject of BASIC's IF-statements, I may as well confess that I have been doing some BASIC programming on an OSI machine recently. The lack of an else-clause has warped my mind to the point that if I had to give a spontaneous BASIC translation of the PASCAL code, I would write:

```
100 Y = Y + 1: Z = Z + 1: IF X <> 3 THEN Y = Y - 2: Z = Z - 2
```

This is atrocious, but I have caught myself writing analogous code over and over. Languages shouldn't encourage this sort of thing.

DAY IS DAZZLED

Jim Day, in a letter published in your May-June 1979 issue, proposed yet another version of the program fragment:

```
100 S = 2 * (X = 3) - 1: Y = Y + S: Z = Z + S
```

This proposal prompts the following remarks:

- Once again the problem of BASIC's nonstandardization rears its ugly head. The designers of OSI's BASIC decided, for reasons known only to themselves, that falsity is better

than truth, so that truth is given the value -1 and falsity the value 0. This means that Mr. Day's expression "2 * (X = 3) - 1" will evaluate to -3 when X equals 3, which is not at all what Mr. Day had in mind. His solution is not transportable.

- Mr. Day implies by omission that the same technique could not be used in PASCAL. This of course is not true—we could write in PASCAL:

```
S = 2 * ord (X = 3) - 1: Y = Y + S: Z = Z + S
```

In my mind, the distinct symbol for the assignment operator enhances legibility, and the explicit presence of the "ord" function in the PASCAL program forces the programmer to be clear that what he is doing is transforming a boolean value into an integer one—this is disguised in the BASIC version. Furthermore, PASCAL is committed to the idea that truth is greater than falsity; ord (true) is always equal to 1 and ord (false) is always equal to 0, so that the PASCAL segment above will run on every implementation I know of.

- Mr. Day's proposal is just a programming trick which happens to work in the case at hand, but it is not capable of general application and utterly fails to address itself to the issue of the else-clause, which was obviously what Mr. Felsenstein had in mind. How would Mr. Day handle this?

```
if X = 3 then theta = sin(x) else beta = cos(y)
```



RAVN-JENSEN IS RETIRED

• I turn now to the article entitled "Round 2" in your May-June 1979 issue. I feel personally involved here, since the PASCAL program you reprinted was one I wrote nearly two years ago. Looking back at it now, I regret that I let my obsession with compactness override my concern for legibility, and wish I had fed it through a prettyprinter before its submission. But that is neither here nor there. I would like to make the following comments on "Round 2":

• To call the language which Mr. Ravn-Jensen uses "BASIC" is pure sophistry. Borge Christensen was at least more candid in his article in your Jan-Feb 1979 issue, where he baptized the language "COMAL", which sounds more like "PASCAL" than like "BASIC". Mr. Ravn-Jensen's program would no more run through a TRS-80, an APPLE, a PET or an OSI interpreter (to name only a few) than my own program would.

• Imitation is the sincerest form of flattery, and given the superiority of PASCAL, it would have been surprising if someone had *not* tried to incorporate some of its features into a new BASIC hybrid. But the fundamental question is, "Why bother?" Is there *anything* in COMAL which Mr. Ravn-Jensen can honestly claim is superior to PASCAL's equivalent? I think not. COMAL supporters will hasten to point out that existing BASIC programs will run on a COMAL interpreter, but is that really important? Every PASCAL system that I know of (the UCSD system, for example) comes with a BASIC compiler or interpreter to take care of existing BASIC programs: so much for that argument. As for rewriting old programs, I claim that it would be no easier to convert them into elegant COMAL than it would be to translate them into PASCAL.

To be sure, any PASCAL supporter must see the spread of PASCAL's features into other languages as a net gain for programming languages as a whole. Given the choice between COMAL and BASIC, I would obviously prefer the former—it has copped enough from PASCAL to be almost tolerable as a beginning programming language. But that is an argument in favor of PASCAL, not against it.

• Mr. Ravn-Jensen violated one of the ground rules of my PASCAL/BASIC comparison: the omission of comments. My intention was to demonstrate that PASCAL is inherently more self-documenting than BASIC, and his program's REM statements make the comparison more difficult.

• Mr. Ravn-Jensen disingenuously remarks that the procedure TALLY is "not needed." I suspect he had an even stronger motive for leaving it out: it is impossible to write such a procedure in COMAL (or in BASIC). This was precisely my point in including the procedure to begin with, in spite of the fact that it is only of marginal value in and of itself. One must assume that programmers will at some point have to write programs more complex than the simple Mastermind program used for illustration, and then the lack of parameter-passing in BASIC becomes absolutely crippling. Suppose one wants to write a fifty-line subroutine which operates on six different variables. To do this in BASIC, one must first of all set aside six global variables (for example, u, v, w, x, y, and z) which are not used by the rest of the program, and then write the following code:

```
100 U = A:V = B:W = C:X = D:Y = E:Z = F:GOSUB 5000
110 A = U:B = V:C = W:D = X:E = Y:F = Z
120 U = G:V = H:W = I:X = J:Y = K:Z = L:GOSUB 5000
130 G = U:H = V:I = W:J = X:K = Y:L = Z
```

Compare this to the same thing in PASCAL:

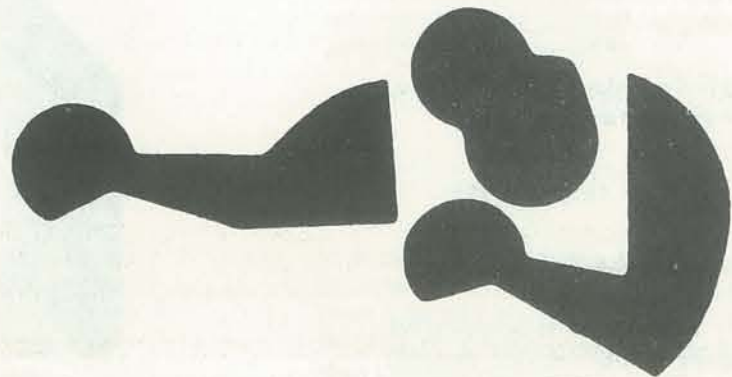
```
process (a, b, c, d, e, f); process (g, h, i, j, k, l);
```

I suppose this strikes me as BASIC's single greatest weakness.

• One must not lose sight of the fact that my Mastermind program deliberately avoided using any of PASCAL's more advanced data structuring facilities—records, pointers, and files. When someone comes out with an 8K COMAL-IN-ROM machine for \$600 which includes those features, I will be the first on my block to order one. Until then, I'll stick with my Microengine.

ROUND 3 IS OVER

There is more that could be said, but I hope that I have at least shown that it is not fair to accuse PASCAL supporters of deliberately choosing inferior dialects of BASIC for their comparisons. Most PASCAL lovers are deeply committed to portability and standardization. It is not our fault that BASIC dialects have proliferated so wildly that there exists no standard BASIC to compare with PASCAL.



TRS-80:

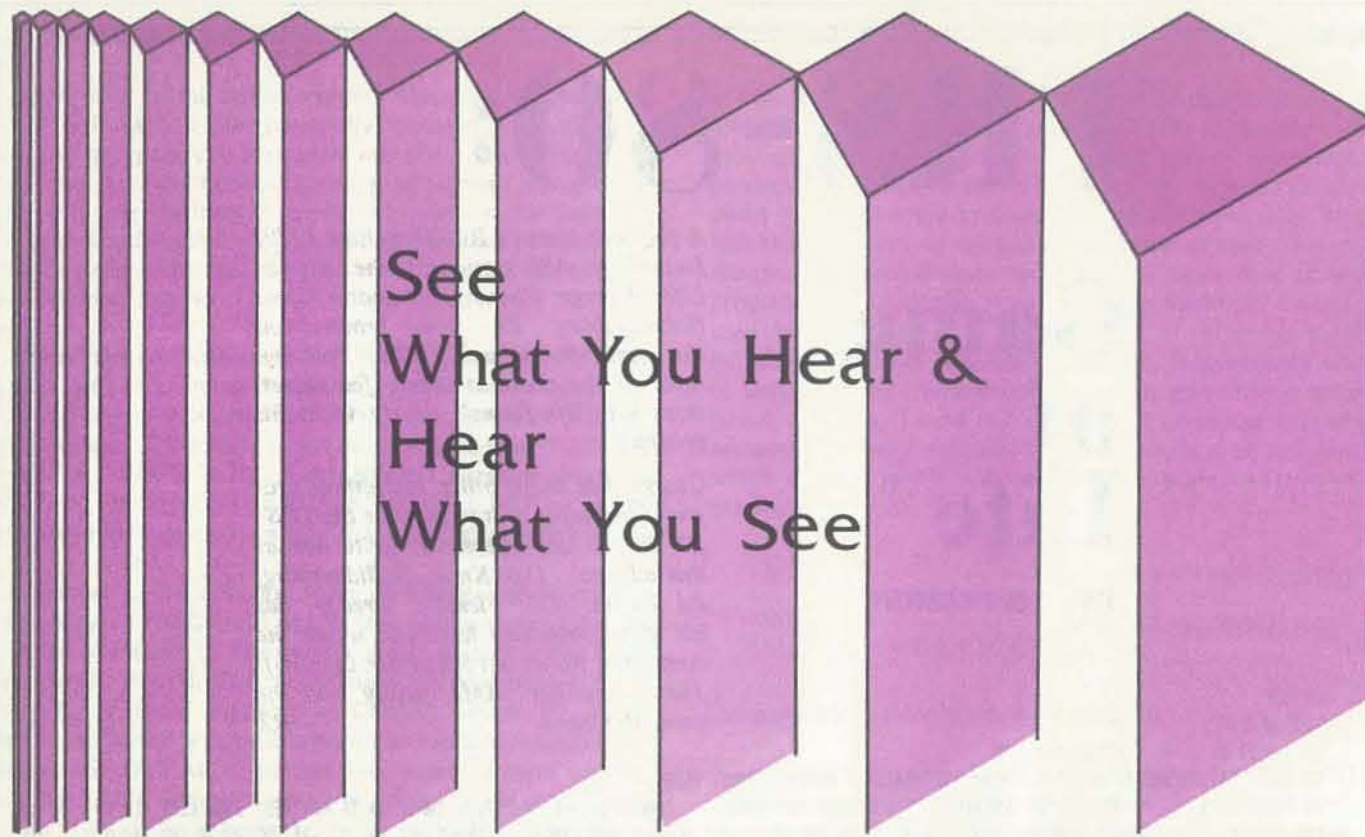
Game of Life

BY G.E. FLEMING

A previous issue of RC (May-June 1979) featured an APL version of the Game of Life. George Fleming, a doctor from Hollidaysburg, PA now brings you the game for your TRS-80. You can refer to the previous article for suggestions on "life-forms" to try with this program.

George has some other interesting programs including a Star Trek for the TRS-80. If you are interested, write him at this address: Oak Knoll, Hollidaysburg, PA 16648. Dr. Fleming already has his kids composing letters to us on the computer. Watch out Milan (Dr. Chepko)! There's another AMA family into the computer game. — RZ

```
10 REM
GAME OF LIFE
DEVELOPED BY G. E. FLEMING 11/17/78
50 CLS: CLEAR 70: DEFINT A, I, J, K, X, Y: D$=STRING$(64, " ") : DIM X(205), Y(205)
90 PRINT CHR$(23): PRINT@128, "DO YOU WANT INSTRUCTIONS FOR " : PRINT " CONWAY'S GAME OF LIFE": INPUT A$: IF A$="YES" 1000 ELSE CLS
100 PRINT@0, " INPUT X, Y AS POSITION. 0,0 IS THE CENTER OF THE SCREEN THE LIMITS OF INPUT ARE +21 TO -21 99,99 TO END INPUT
ENTER": INPUT A$: PRINT@ 0,0$:D$
110 PRINT@0,0$:D$: PRINT@0, "X=? Y=?": INPUT X, Y
120 IF X=99 AND Y=99 THEN 200 ELSE IF ABS(X)>21 OR ABS(Y)>21 THEN 100
130 X=2*X+64 Y=Y+23
140 IF POINT(X, Y) PRINT@0, "YOU ALLREADY USED THAT LOCATION PLEASE TRY AGAIN." FOR J=1 TO 500: NEXT
150 IF NOT POINT(X, Y) SET(X, Y): SET(X-1, Y): T0TAL=T0TAL+1: X(T0TAL)=X: Y(T0TAL)=Y
160 GOTD110
200 CLS: IF T0TAL>MAX THEN MAX=T0TAL
210 PRINT@0, PRINT USING "GEN ###": GEN: GEN=GEN+1: PRINT@64, PRINT USING "POP ###": T0TAL: PRINT@128: PRINT USING "BIRTH ###": BIRTH:
BIRTH=0: PRINT@192: PRINT USING "DEATH ###": DEATH: DEATH=0: PRINT@256: PRINT USING "MAX ###": MAX:
220 IF T0TAL=0 FOR G=1 TO 500: NEXT: CLS: PRINT CHR$(23): PRINT@130, "POPULATION 0, END", " GENERATION " GEN, " MAX POP. WAS " MAX: END
230 XUP=X(1): XDN=X(1): YUP=Y(1): YDN=Y(1)
240 FOR J=1 TO T0TAL: IF X(J)>XUP THEN XUP=X(J)
241 IF X(J)<XDN THEN XDN=X(J)
242 IF Y(J)>YUP THEN YUP=Y(J)
243 IF Y(J)<YDN THEN YDN=Y(J)
244 NEXT
250 IF XDN<=3 OR XUP>=124 OR YDN<=1 OR YUP>=46 THEN CLS: PRINT CHR$(23): PRINT@128, "LIMITS EXCEEDED - GAME OVER": PRINT@266, "GENERATION"
GEN: PRINT@390, "MAX POP. WAS " MAX: END
255 FOR J=1 TO T0TAL: SET(X(J), Y(J)): SET(X(J)-1, Y(J)): NEXT: T0TAL=0
260 FOR X=XDN-2 TO XUP+2 STEP 2: FOR Y=YDN-1 TO YUP+1: IF POINT(X, Y) THEN RESET(X, Y) ELSE SET(X, Y)
270 A1=POINT(X-2, Y-1) A2=POINT(X, Y-1) A3=POINT(X+2, Y-1) A4=POINT(X-2, Y) A5=POINT(X, Y) A6=POINT(X+2, Y) A7=POINT(X-2, Y+1) A8=POINT(X, Y
+1) A9=POINT(X+2, Y+1)
280 IF (A5=0) AND (A1+A2+A3+A4+A6+A7+A8+A9=-2) THEN T0TAL=T0TAL+1: X(T0TAL)=X: Y(T0TAL)=Y: GOTD320
290 IF (A5=0) AND (A1+A2+A3+A4+A6+A7+A8+A9=-3) THEN T0TAL=T0TAL+1: X(T0TAL)=X: Y(T0TAL)=Y: GOTD320
300 IF (A5=-1) AND (A1+A2+A3+A4+A6+A7+A8+A9=-3) THEN T0TAL=T0TAL+1: X(T0TAL)=X: Y(T0TAL)=Y: BIRTH=BIRTH+1: GOTD320
310 IF NOT POINT(X, Y) DEATH=DEATH+1
320 IF T0TAL=250 CLS: PRINT CHR$(23): PRINT@128, "POPULATION LIMIT EXCEEDED", " GEN. " GEN, " POP. = 250": END
330 IF POINT(X, Y) THEN RESET(X, Y) ELSE SET(X, Y)
340 NEXT Y, X: GOTD200
1000 CLS: PRINT CHR$(23): PRINT@134, "THE RULES OF THE GAME ARE IF A CELL HAS 2 OR THREE NEIGHBORS IT STAYS ALIVE OTHER WISE IT DIES. I
F AN EMPTY CELL HAS EXACTLY 3 NEIGHBORS IT IS BORN. ALL THE BIRTHS AND DEATHS OCCUR AT THE SAME TIME. ENTER FOR AN EXAMPLE ":
1010 INPUT A$: T0TAL=7: FOR I=1 TO 7: READ X(I): READ Y(I): NEXT: DATA 64, 23, 64, 24, 64, 25, 66, 23, 68, 23, 68, 24, 68, 25: GOTD200
```

See
What You Hear &
Hear
What You See

BY HERB MOORE

Who is Herb Moore? Perhaps you have seen this query in some of the recent FuturePlay™ sections of RC. Well, Herb is a musician and beginning computer person. He has become interested in computers since some manufacturers have begun including a sound chip with the new machines.

Herb is a student and innovator in an emerging field of music call "scrapophony." Scrapophony involves the use of found materials to create musical instruments. Some of Herb's recent compositions (available on cassette tape) are performed on "scrapophones" that he has constructed. You will no doubt "hear" more from Herb in coming issues. The way computers are going, we believe that Herb may turn out to be our first "music editor."
—RZ

This is the first in a series of articles describing some of the color graphics and sound capabilities of the new ATARI computers. The articles, if read in sequence, should be easily understood by people who have had little previous experience in BASIC programming. The first article covers the color graphics available on the machines. The second will describe the sound capabilities, and in later articles we will show how the color graphics and sound can be combined, giving you the opportunity to "see what you hear and hear what you see."

Sections of this article were excerpted from the book *ATARI BASIC* by Bob Albrecht, LeRoy Finkel, and Jerald R. Brown, ©1979, by Wiley & Sons, Inc. (A book worth having if you have just purchased an ATARI computer or are thinking about it.)

But first, let's "see what you can see!" The ATARI machine allows you to:

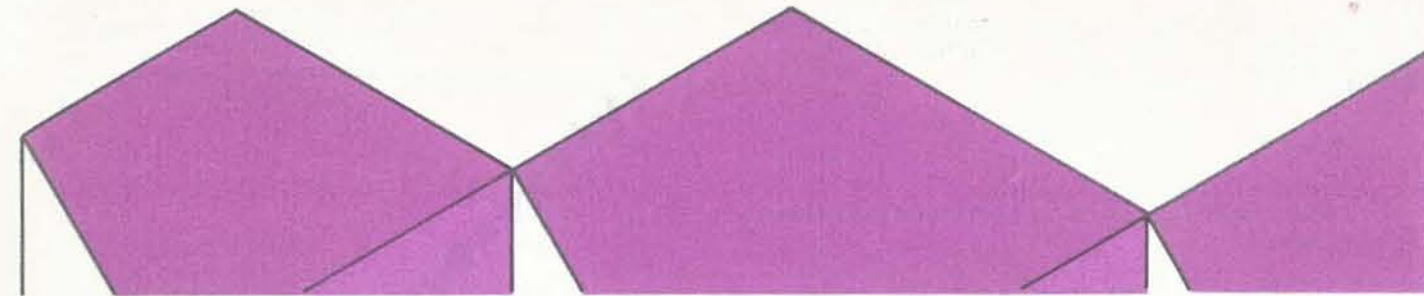
- change the background color of the video screen to any one of 16 colors;
- draw designs on the video screen using any one of 16 colors;
- print text along with your pictures in a special text 'window';
- change the color of the text window to any one of 16 colors;
- change the number of plotting positions so that pictures can be drawn in more detail.

ATARI GRAPHICS MODES

The ATARI machine also has 8 different graphics modes, numbered 0-7. When the machine is turned on, the screen mode is automatically set to GRAPHICS 0. The different graphics modes allow you to put finer and finer detail on the screen.

In order to select a different mode, you would type in either GRAPHICS or GR. and the number of the mode you wanted to use. For example, typing in the command:

```
GRAPHICS 3
```



would set the machine to graphics mode 3. Typing in the command:

```
GR. 3
```

(be sure to use the period)

would also set the machine to graphics mode 3. Here's what happens to the screen when we enter this command. It causes the machine to:

- Blank the top portion of the screen, and
- Color the bottom portion purple (this color may vary with the TV set you use).

Graphics mode 3 provides a grid of 39 columns by 20 rows for plotting points. This is the blank area at the top of the screen. The purple area at the bottom of the screen provides four lines in which text may be printed. To return to the text mode you type:

```
GRAPHICS 0
```

and press the RETURN key

Let's write a simple program which puts labels on the four lines in the text window.

First type GRAPHICS 0 to return to the text mode. Then type NEW (this clears the memory of the machine of any other information that might become confused with your new program). Now type in the following program:

```
10 GRAPHICS 3
20 PRINT "LINE 1"
30 PRINT "LINE 2"
40 PRINT "LINE 3"
50 PRINT "LINE 4";
60 GOTO 60
```

(Note: don't forget the semicolon)

Here's what you told the machine to do:

- The line numbers 10, 20, etc., indicate that you are entering a program. If you don't use line numbers the machine will see the information as DIRECT COMMANDS and execute them immediately.
- The PRINT statements in lines 20 to 50 tell the machine to print what is between the quotation marks.
- The semicolon at the end of line 50 will cause the cursor to remain at the end of the message when the program is run.
- Line 60 tells the machine to wait while the information is displayed on the screen.

In order to run the program, you type the command RUN and press the ENTER key. This program, when run, would look like this at the bottom of the screen:

```
LINE 1
LINE 2
LINE 3
LINE 4 ■
```

Here is a table of the graphic screen modes 3 through 7, showing the number of graphics positions for each mode.

MODE NO.	NUMBER OF COLUMNS	GRAPHIC POSITIONS ROWS	TEXT WINDOW
3	39	20	4 lines
4, 5	79	40	4 lines
6, 7	158	80	4 lines

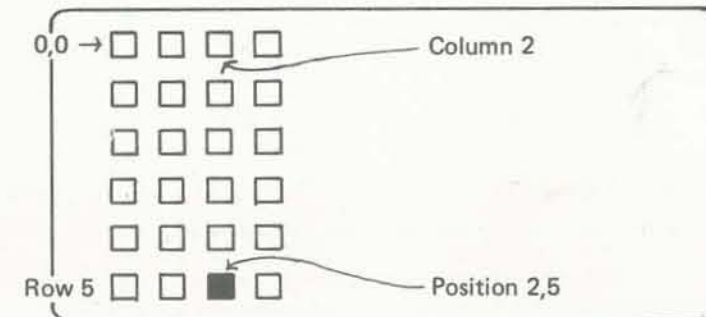
From this table we can see that modes 6 and 7 have 158 × 80, or 12,640 individual points. Modes 4 and 5 have 79 × 40, or 3,160 points. Mode 3 has 39 × 20, or 780 points. Thus, for example, graphics modes 6 and 7 allow you to draw with smaller points, enabling greater detail. If you wanted to cover larger portions of the screen with less programming, then you would probably want to use graphics mode 3.

PLOT AND COLOR

The PLOT statement is used for graphics on the ATARI computer. To plot a point, both the column and the row must be given in the PLOT statement. The plot positions are numbered from the upper-left corner of the screen starting with 0,0. For example, a plot statement of:

```
30 PLOT 2,5
```

would plot the following point.



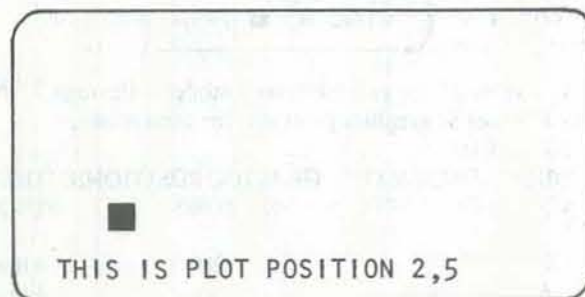
In order to see the point, we need to put a color statement in the program for one of the 4 available color registers, numbered 0-3. For example:

```
COLOR 1
```

Using the information given so far, you can write the following short program:

```
10 GR. 3
20 COLOR 1
30 PLOT 2,5
40 PRINT "THIS IS PLOT POSITION 2,5"
```

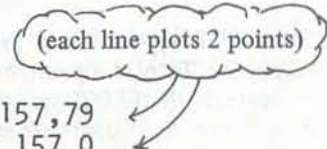
The program, when run would look like this on the screen.



It plots the point on the screen in the graphic area. It prints in the text window the information that is between the quotation marks in the PRINT statement.

If we wanted to plot several points, we could include several plot statements on one line by using a colon to separate the statements. First, type GRAPHICS 0 to return to the text mode, then type NEW to begin a new program. Then enter and run the following program:

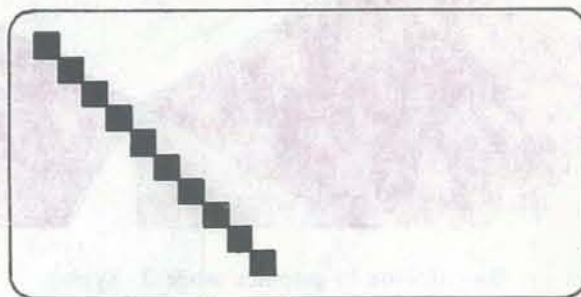
```
10 GR. 6
20 COLOR 1
30 PLOT 0,0: PLOT 157,79
40 PLOT 0,79: PLOT 157,0
```



If we wanted to draw a straight line we wouldn't want to make a PLOT statement for every point of the line. Here's a couple of programming time savers we can use. The following program makes use of the FOR-NEXT loop to plot points. Type NEW, enter and run the following:

```
10 GRAPHICS 3
20 COLOR 1
30 FOR I=0 TO 19
40 PLOT I,I
50 NEXT I
```

When the program is run, the screen looks like this:



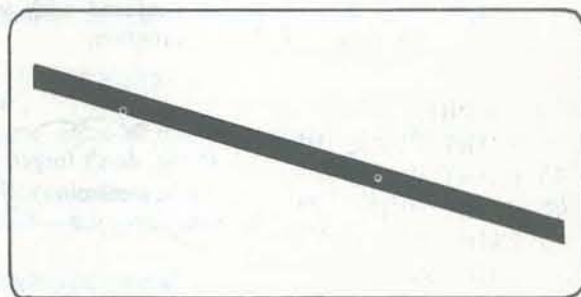
The instruction in line 30 tells the machine to give the letter I a value from 0 to 19, beginning with 0. Line 40 tells the machine to plot a point for the given value of I. Line 50 tells the machine to go to the next value of I. So the machine plots 0,0 then 1,1 then 2,2 and so on.

DRAWTO STATEMENT

The easiest way to plot a straight line is to use the DRAWTO statement along with the PLOT statement. The PLOT statement tells the computer where to start the line, and the DRAWTO statement gives the position where the line will end. The line is drawn from the PLOT position to the DRAWTO position. For example, if we enter and run the following program:

```
10 GR. 4
20 COLOR 1
30 PLOT 0,0
40 DRAWTO 78,39
```

The screen would show:



SET COLOR STATEMENT

The statement used to change color points on the screen is:

```
SET COLOR 0,C
```

The variable C, can be any integer from 0 through 15. Each value for C will give us a different color for the plotted points.

A complete color graphics program must have the three statements: GRAPHICS, COLOR, and SETCOLOR. Of course, you may also plot some points. For example, type NEW, enter and run the following:

```
10 GR. 3
20 COLOR 1
30 SETCOLOR 0,0
40 PLOT 0,3
50 DRAWTO 38,3
60 PRINT "WHAT COLOR DO YOU SEE?"
```

A long grey color bar (color may vary for your TV) appears in the top portion of the screen, and the statement WHAT COLOR DO YOU SEE? appears at the bottom of the screen. A few changes and additions converts this program to show each of the 16 colors available. The colors are changed by the values of the variable in the SETCOLOR statement in line 30. If we enter and run the following program:

```
10 GR. 3
20 COLOR 1
25 FOR N = 0 TO 15
30 SETCOLOR 0,N
40 PLOT 0,3
50 DRAWTO 38,3
60 PRINT: PRINT: PRINT: PRINT N
70 FOR W = 1 TO 500: NEXT W
80 NEXT N
```

a bar appears at the top of the screen, disappears and then reappears in a different color, for the 16 different colors available on the ATARI computer.

If we revise line 30 in this program to read:

```
30 SETCOLOR 2,N
```

and run the program, the color bar remains gold throughout the run, but the color of the text window changes to each of the 16 colors.

If we make one last change in line 30 of the program:

```
30 SETCOLOR 4,N
```

and run the program, the background of the graphics area changes through the 16 colors, the bar remains gold, and the text window turns dark and remains dark.

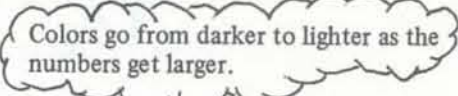
So, for the statement SETCOLOR A,N:

- When A is set to 0, the colors in the plotted bar vary.
- When A is set to 2, the colors in the text window vary.
- When A is set to 4, the colors in the background of the graphics area vary.

In the same statement the variable N allows for the selection of one of 16 colors (0-15) for a given color register.

Here is a table of the 16 colors available:

- 0 Grey
- 1 Gold
- 2 Orange
- 3 Red
- 4 Pink
- 5 Purple
- 6 Purple
- 7 Blue
- 8 Blue
- 9 Blue
- 10 Blue
- 11 Green
- 12 Green
- 13 Green
- 14 Yellow Green
- 15 Gold



These effects can all be shown by entering and running the following program.

```
10 GR. 3
20 COLOR 1
30 PRINT "WATCH MY COLORS CHANGE! ! !!"
40 FOR M = 0 TO 4 STEP 2
50 FOR N = 0 TO 15
60 SETCOLOR M,N
70 PLOT 5,5: DRAWTO 30,5
80 FOR W = 1 TO 200: NEXT W
90 NEXT N
100 NEXT M
```

When this program is run, first the color bar varies through 16 colors, then the text window varies through 16 colors, and finally the background of the graphics area varies through 16 colors. The command STEP 2 in line 40 tells the machine to count by 2's. So, it counts 0, 2, 4.

MORE GRAPHICS

In order to see what happens as you vary the graphics mode, enter the following program.

```
10 GR. 3
20 COLOR 1
30 SETCOLOR 4,1
40 PLOT 5,5: DRAWTO 7,5
50 PLOT 6,6: DRAWTO 6,10
60 DRAWTO 4,15
70 PLOT 6,10: DRAWTO 8,15
80 PLOT 5,7: DRAWTO 3,9
90 PLOT 7,7: DRAWTO 9,9
100 PRINT "WHAT A FUNNY MAN! ! !!"
```

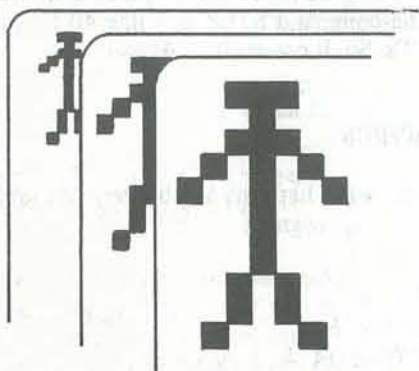
When run, this program causes a figure, that looks like this:



to appear in the upper left hand portion of the screen. If you make the following changes and additions to the program to alter the screen modes, you can change the size of the "drawn" figure:

```
5 FOR A = 7 TO 3 STEP-2
10 GRAPHICS A
93 FOR W = 1 TO 200: NEXT W
95 NEXT A
```

When this program is run, the figure appears to move from the upper left corner, and his size increases. The command STEP-2 causes the program to change the graphics mode from 7 (many small divisions), to graphics mode 5 (fewer, slightly larger divisions), to graphics mode 3 (fewer, and even larger divisions). This result is an example of something that was pointed out earlier—in graphics mode 7, it is possible to draw in greater detail since you use smaller points; in graphics mode 3, it is possible to cover larger areas of the screen with fewer plotted points.

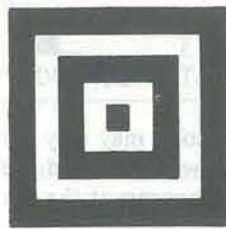


Here is another example which allows you to examine different sized squares. First, enter and run the following program.

```
10 GRAPHICS 7
20 COLOR 1
30 SETCOLOR 0,15
35 X = 80:Y = 48
40 PLOT X,Y
50 PLOT X-2,Y-2: DRAWTO X+2, Y-2
60 DRAWTO X+2, Y+2: DRAWTO X-2,Y+2
```

```
70 DRAWTO X-2,Y-2
80 PLOT X-4,Y-4: DRAWTO X+4,Y-4
90 DRAWTO X+4,Y+4: DRAWTO X-4,Y+4
100 DRAWTO X-4,Y-4
```

This should cause the machine to draw two squares, one inside of the other.



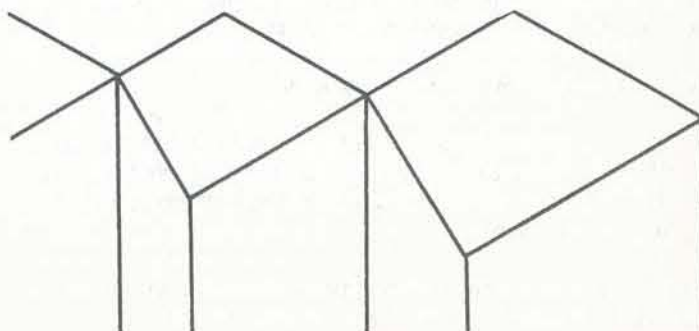
If we make changes, the size of the square and the width of the lines get larger each time.

```
10 GRAPHICS 5
35 X=40: Y=24.
10 GRAPHICS 3
35 X=20: Y=12.
```

You can create an interesting visual effect by combining the last three graphics modes in one program. The resulting program causes two squares to appear in the center of the screen, one inside of the other. They should then become increasingly larger with increasingly thicker lines until the program has gone through the three different modes. Then it returns to the smaller squares with smaller lines and begins to go through the cycle again. This visual pattern will continue on the screen until you stop the program by pressing the BREAK key.

This should give the reader a taste of some of the color graphics capabilities of the ATARI computer. If you are fortunate enough to have an ATARI computer already, this material will help you begin to create your own color graphics effects. Please be sure to let us know of any interesting discoveries you make in your experiments with the machine.

Next time: Atari Music



SUBSCRIPTION ORDER

Please send me a one-year subscription to **Recreational Computing** magazine, formerly **People's Computers** (published bi-monthly) for \$10.

NAME _____

ADDRESS _____

CITY/STATE _____ ZIP _____

Check enclosed Bill me (U.S. only)
 Renewal (please attach mailing label)

Charge my card: Visa/BankAmericard Mastercharge
Card No. _____ Exp. Date _____

Signature _____

International Rates: Canada *First Class* \$17 One Year
Rest of World *Airmail* \$23 One Year

World Surface Mail — I will risk the lengthy and unreliable delivery of surface mail (signed) _____

\$14 One Year
Payment must be in \$US drawn on a US Bank.

A 3

Exp. Date: 12/31/79

Exp. Date: 12/31/79

demonstration Pascal source and object programs, translator, Pascal library, and three run-time systems. People's Pascal runs on any 16K Level II TRS-80 system, \$15 (\$13.50 to TRS-80 Computing subscribers)*.

TAPE 4 LEVEL I:

Election returns, business percentage, math problems, Snoopy graphic, cash register, action on the Enterprise, chase, commander-in-chief, Christmas graphic, ups & downs of business, deep dark secrets, air raid, cross index, speech recording aid, inventory control, check-book balance, sales receipt tally, \$7.50*.

TAPE 5 LEVEL II:

Executive tension-breaker, lineprinter to screen or screen to lineprinter exchange utility program, election returns, business percentage, check balancing, car pool database, RAM memory test, mind bender, mortgage payments, spelling bee, common factor, diet planning II, Star Trek III, Star Wars, klindon capture, knights, federal income tax, chase, tachscope, music ("Yesterday"), vacation planner, \$7.50*.

*ADD 50¢ postage, handling; CA residents add tax.

DEALER INQUIRIES INVITED

grams, half of which came from the San Diego TUG (user group), required many more hours preparation.

People's software was not brought out to compete with commercial software. CIE encourages TRS-80 users to market programs. The nonprofit organization gives free space in TRS-80 Bulletin, as well as offering low-cost Bulletin ads. Good commercial software is the foundation of our TRS-80 computing, starting with TRS-80's Micro Soft Level II Basic ROMs.

With introduction of People's Software, personal computing can be made much more rewarding for everyone, since users can share the fruits of their labor—and this can be inexpensively distributed, in machine-readable form (no \$0.0433-per-hour keyboarding!).

Digital Equipment Corp (DEC) computer users, by joining DECUS, have been able to share programs inexpensively. Now Radio Shack users have People's Software.

computer information exchange, inc.

box 158 san luis rey ca 92068

-- PEOPLE'S SOFTWARE --

- amt. encl.
- () Tape 1 Level II, \$8* () Tape 1 Level I, \$8*
 - () Tape 2 Level II, \$8* () Tape 3 Level II, \$15.50 (\$14, "TRS-80 Computing" subscribers)
 - () Tape 4 Level I, \$8*
 - () Tape 5 Level II, \$8* *includes 50¢ postage, handling

Make checks payable COMPUTER INFORMATION EXCHANGE

Charge my VISA (), MasterCard () # _____

_____(signed), expires _____

name _____

street address _____

city, state, zip _____

SOFTWARE: m tape \$7.50

on't have to shell out a bundle of cash to get a good assortment of the world's most popular computer. Gives you up to 77 public-domain software tapes, just \$7.50 plus 50¢ (CA residents add 45¢ tax—be paid in U.S. funds; postage

Information Exchange has been public domain software can be made available to the public.

Library of computer magazines can be made available at no more cost than a subscription. CIE's experience has shown that the 77-program Tape 2. As a result, the \$7.50 selling price of the software (the tape), a person doing the math is paying 4.33 cents per programming hour, though it contains fewer pro-

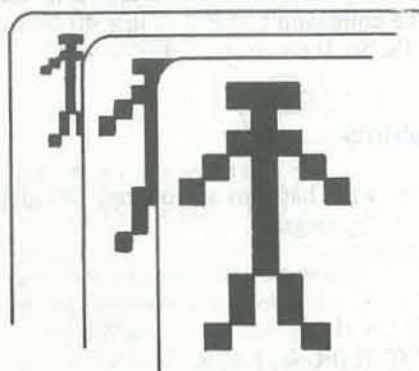
When run, this program causes a figure,



to appear in the upper left hand portic make the following changes and addit alter the screen modes, you can change figure:

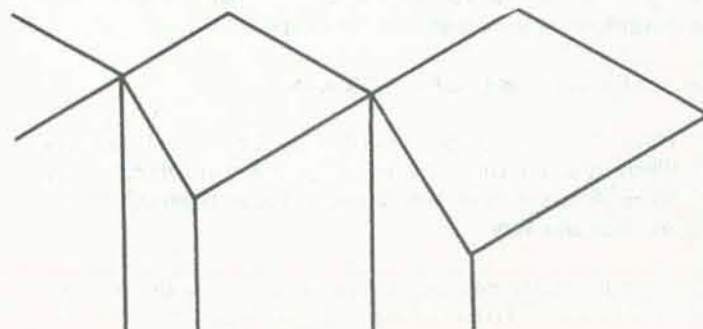
5 FOR A = 7
10 GRAPHICS
93 FOR W = 1
95 NEXT A

When this program is run, the figure appears to move from the upper left corner, and his size increases. The command STEP-2 causes the program to change the graphics mode from 7 (many small divisions), to graphics mode 5 (fewer, slightly larger divisions), to graphics mode 3 (fewer, and even larger divisions). This result is an example of something that was pointed out earlier—in graphics mode 7, it is possible to draw in greater detail since you use smaller points; in graphics mode 3, it is possible to cover larger areas of the screen with fewer plotted points.



Here is another example which allows you to examine different sized squares. First, enter and run the following program.

```
10 GRAPHICS 7
20 COLOR 1
30 SETCOLOR 0,15
35 X = 80:Y = 48
40 PLOT X,Y
50 PLOT X-2,Y-2: DRAWTO X+2, Y-2
60 DRAWTO X+2, Y+2: DRAWTO X-2,Y+2
```



Next time: Atari Music

This should give the reader a taste of some of the color graphics capabilities of the ATARI computer. If you are fortunate enough to have an ATARI computer already, this material will help you begin to create your own color graphics effects. Please be sure to let us know of any interesting discoveries you make in your experiments with the machine.

You can create an interesting visual effect by combining the last three graphics modes in one program. The resulting program causes two squares to appear in the center of the screen, one inside of the other. They should then become increasingly larger with increasingly thicker lines until the program has gone through the three different modes. Then it returns to the smaller squares with smaller lines and begins to go through the cycle again. This visual pattern will continue on the screen until you stop the program by pressing the BREAK key.

35 X=40: Y=24.
10 GRAPHICS 3
35 X=20: Y=12.

DR. DOBB'S JOURNAL of
COMPUTER
Calisthenics & Orthodontia

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 756 MENLO PARK, CA

POSTAGE WILL BE PAID BY ADDRESSEE

DR. DOBB'S JOURNAL
P.O. BOX E
1263 EL CAMINO REAL
MENLO PARK, CA 94025

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

Announcing-

--PEOPLE'S SOFTWARE--

TAPE 1 Level-I: 24 business, educational, entertainment programs, \$7.50.*

TAPE 1 LEVEL II

Additional 10 programs, adds to journal/ledger, payroll, perpetual calendar, Star Treks, etc., touch typing, mail addressing, tic-tac-toe, grand prix auto race, state capital quiz, hangman spelling game, \$7.50*.

TAPE 2 LEVEL II:

Fully documented in "Some Common Basic Programs" by Lon Poole & Mary Borchers (Osborne & Associates), 77 business, investment, mathematical, statistic, and home-use programs, including federal withholding, tax depreciation, check writer, recipe cost, map check, day of week, days between two dates, anglo to metric, alphabetize, etc., \$7.50*.

TAPE 3 LEVEL II:

PEOPLE'S PASCAL program development system, royalty software developed by John Alexander, based on the Kin-Man Chung/Herbert Yuen "A 'Tiny' Pascal Compiler" articles in the September ('78), October and November "Byte".

The system includes editor/compiler, interpreter, demonstration Pascal source and object programs, translator, Pascal library, and three run-time systems. People's Pascal runs on any 16K Level II TRS-80 system, \$15 (\$13.50 to TRS-80 Computing subscribers)*.

TAPE 4 LEVEL I:

Election returns, business percentage, math problems, Snoopy graphic, cash register, action on the Enterprise, chase, commander-in-chief, Christmas graphic, ups & downs of business, deep dark secrets, air raid, cross index, speech recording aid, inventory control, check-book balance, sales receipt tally, \$7.50*.

TAPE 5 LEVEL II:

Executive tension-breaker, lineprinter to screen or screen to lineprinter exchange utility program, election returns, business percentage, check balancing, car pool database, RAM memory test, mind bender, mortgage payments, spelling bee, common factor, diet planning II, Star Trek III, Star Wars, klinton capture, knights, federal income tax, chase, tachitscope, music ("Yesterday"), vacation planner, \$7.50*.

*ADD 50¢ postage, handling; CA residents add tax.

DEALER INQUIRIES INVITED

PEOPLE'S SOFTWARE: 77-program tape \$7.50

Now TRS-80 owners don't have to shell out a bundle of money or work hard to get a good assortment of programs for the world's most popular computer.

People's Software gives you up to 77 public-domain programs on one cassette tape, just \$7.50 plus 50¢ postage and handling (CA residents add 45¢ tax—FOREIGN orders must be paid in U.S. funds; postage is \$1 per tape, via air).

Nonprofit Computer Information Exchange has been concerned about how public domain software can be easily made available to the public.

Anyone with a library of computer magazines can keyboard-in a wealth of software, at no more cost than the user's time and frustration. CIE's experience has been about 150 hours for the 77-program Tape 2. Assuming that \$1 of the \$7.50 selling price of the software is medium cost (the tape), a person doing the job himself will be saving 4.33 cents per programming hour. Tape 1, even though it contains fewer programs, half of which came from the San Diego TUG (user group), required many more hours preparation.

People's software was not brought out to compete with commercial software. CIE encourages TRS-80 users to market programs. The nonprofit organization gives free space in TRS-80 Bulletin, as well as offering low-cost Bulletin ads. Good commercial software is the foundation of our TRS-80 computing, starting with TRS-80's Micro Soft Level II Basic ROMs.

With introduction of People's Software, personal computing can be made much more rewarding for everyone, since users can share the fruits of their labor—and this can be inexpensively distributed, in machine-readable form (no \$0.0433-per-hour keyboarding!).

Digital Equipment Corp (DEC) computer users, by joining DECUS, have been able to share programs inexpensively. Now Radio Shack users have People's Software.

computer information exchange, inc.

box 158

san luis rey ca 92068

--PEOPLE'S SOFTWARE--

amt.
encl.

- () Tape 1 Level II, \$8* () Tape 1 Level I, \$8*
() Tape 2 Level II, \$8* () Tape 3 Level II, \$15.50
() Tape 4 Level I, \$8* (\$14, "TRS-80 Computing" subscribers)
() Tape 5 Level II, \$8* *includes 50¢ postage, handling

Make checks payable COMPUTER INFORMATION EXCHANGE

Charge my VISA (), MasterCard () # _____
(signed), expires _____

name _____

street address _____

city, state, zip _____

Newett Awl and the Goat

BY GORDON FRENCH

PART II: THE SOLUTION

When we left Newett, he was discussing his solution with Gordon and Gordon's neighbor, the part-time policeman. They had just been told that the problem could be solved by looking at the area that the goat eats as two circular segments. The problem, once again, was to write a program that would tell you how long to make a rope so that if you staked a goat at the edge of a 20 foot diameter circle, the goat would eat exactly half the grass.

Figure 1 was Newett's rough sketch of the problem. Figure 2 gives the information needed to set up the equation to calculate a circular segment. The formula is $A = \frac{1}{2}[RL - C(R - H)]$ where R is the radius, L is the arc length, and C is the base.

Here is Newett's program and the rest of the story. Enjoy!!
- RZ

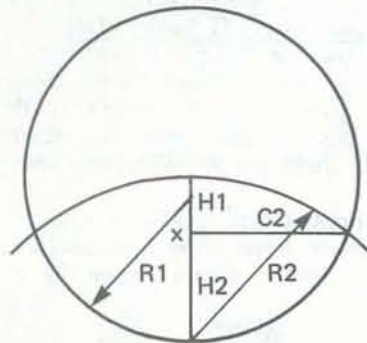


Figure 1

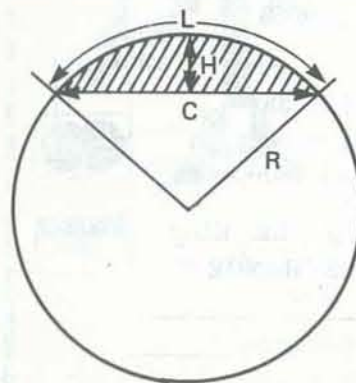
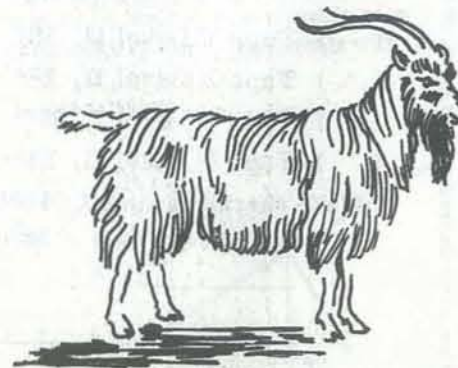


Figure 2

Newett showed us the listing, and began to explain the code line by line. As best as I can remember his comments went something like this:

- Lines 10 and 20 set up two dummy values to make the transcendental functions come out in degrees instead of radians which most computers usually deliver.
- Lines 30 and 40 are the input to the program for the diameter of the lawn and the percentage that the goat is to eat.
- Line 50 sets a limit counter for use later and line 60 calculates the area of the lawn for the diameter given. Line 70 calculates the goat's portion.
- Line 80 thru 95 spill out the calculations and let you know that the problem is being worked on.
- Lines 100 and 110 set the initial angle to 45 degrees and the radius of the lawn to half the diameter.
- Line 120 calculates the length of half of the base of the circular segments. It is common to both segments for this solution.
- Line 130 calculates a factor later used to determine the angle for one of the two circular segments.
- Line 140 calculates the height of the outer segment.
- Line 150 calculates the radius of the inner segment and is really the rope length.
- Line 160 calculates the height of the inner segment.
- Line 170 calculates the common base for both segments.
- Line 180 calculates the angle of the outer segment.
- Line 190 sets up the parameters for the outer segment and calls the subroutine that calculates the area for the outer segment.



- Line 200 saves the area of the outer segment as the variable A2.
- Lines 210 thru 230 do the same for the inner segment.
- Line 240 combines the two areas for testing.
- Line 250 tests if the goat's area is greater than the area of the two segments and, if it is, directs the program to increase the angle and recalculate the two areas.
- Line 260 tests if the goat's area is less than the area of the two segments and, if it is, directs the program to decrease the limit factor by half, and decrease the angle and recalculate the two areas. Newett commented that the ELSE statement in the Line 260 will end the program if the two areas are exactly equal, but said that it is improbable that the program would terminate there.
- Lines 270 and 280 decrease the limit factor and the angle.
- Line 300 increases the angle for the next approximation.
- Line 310 tests if the limit factor is still significantly large

to effect the angle. Newett explained that this is the means he used to detect if the angle had been calculated to the limits of accuracy of the BASIC being used. This is the test that will most often limit the calculations and get you out of the loop.

- Lines 320 and 330 calculate the length of the arc and the area of the segment whose parameters were passed to the routine as Y, R, H, and C.
- Line 350 spills the result of the calculations.

I was really quite pleased, and so was Newett. While I was busy explaining how clever all this was to my neighbor, Newett began to roll himself a smoke. Yep! Out came the badge and the cuffs and before I could think, Newett was hauled away. I'd have gone down to bail him out, but the microbus is locked up and neatly parked in front of my driveway.

Listing

```

10 J=1.745329252E-02
20 J1=1/J
30 INPUT"DIAMETER OF LAWN = ",D
40 INPUT"PERCENTAGE OF LAWN GOAT IS TO EAT = ",P
50 F=1
60 A=3.14159265*(D/2)^2
70 G=A*(P/100)
80 PRINT"AREA OF LAWN IS ",A," SQUARE UNITS"
90 PRINT"GOAT'S PART IS ",G," SQUARE UNITS"
95 PRINT"CALCULATING. . . ."
100 N1=45
110 R1=D/2
120 C2=SIN(N1*J)*R1
130 X=SQRT(R1^2-C2^2)
140 H2=R1-X
150 R2=SQRT(H2^2+C2^2)
160 H1=R2-H2
170 C=C2*2
180 Y=2*J1*ATN(C2/H2)
190 H=H2 \ R=R1 \ GOSUB 320
200 A2=A
210 Y=2*J1*ATN(C2/X)
220 H=H1 \ R=R1 \ GOSUB 320
230 A1=A
240 A=A1+A2
250 IF G > A THEN 300
260 IF G < A THEN 270 ELSE 350
270 F=F/2
280 N1=N1-F
290 GOTO 310
300 N1=N1+F
310 IF N1 + F = N1 THEN 350 ELSE 120
320 L = J * R * Y
330 A = 1/2 * (R * L - C * (R - H))
340 RETURN
350 PRINT"MAKE THE ROPE ",R2," UNITS LONG"
360 END
    
```



^ → raising to a power.
\ → multiple statements.
See page 71 of this issue for a reader response to this challenge.



Gandalf Comes to Madison High

BY RALPH ROBERTS

Photos of visitors to the May science fair in the Asheville Mall interacting with Gandalf, by the author.

If kids want to compute, they should have the chance. So reasoned Ralph Roberts, who recently helped a group of North Carolina high school students put together a computer from a kit. Ralph, who describes the birth of the computer in the story below, is known to many of you for his RC fiction (his latest for us — "What Light on Yonder Panel Flashes?" in the July-August issue). —LB

Madison High sits on a mountaintop overlooking the small town of Marshall, in the Blue Ridge mountains of Western North Carolina. Surrounded by higher peaks of unsurpassed beauty, this setting seems straight out of Tolkien. You can almost see cloud-enshrouded castles, flying dragons, and (look closely) a hobbit or two. Indeed, it's a most fitting setting for the reincarnation of the mighty Gandalf.

And what form does the great wizard achieve in this life? A computer, naturally. To be more specific: a SWPTC MP6800, lovingly constructed from a kit by the students of the high school math and electronics clubs. The system also includes an old but still sturdy ASR33 teletype (formerly of Western Union); a couple of Smoke Signal floppy drives; a home-built terminal (parts donated by a local microswitch manufacturer); and a custom cabinet.

Three teachers at Madison High—Dave Cox, Rex Sprinkle, and Louis Zimmerman—oversaw and assisted in Gandalf's rebirth. They worked long and hard in the raising of funds, the procurement of parts, and the finagling of outside help (I was one of the helpers).

The result was a very good computer system, owned not by the school, but by the students. A computer that opens up a magical new world of knowledge and allows 'hands-on' experience.

Gandalf's coming-out party was at the May science fair in the nearby Asheville Mall. Here, Gandalf calculated biorhythms, played a stock market game, and performed musical selections for the awed passersby. In general, he was the center of attention, and the students of Madison High received well-deserved acclaim.



Designing Animal Games

BY MIKE GABRIELSON AND TOM MARTIN

Does your cat have three legs? Is an octopus with seven arms a septipus? If you are designing your own Animal game, you can have any creatures you want, even mythical beasts. In fact, the design hints for this game can be used to produce many other "learning" programs.

The next step is for someone to modify the program to produce the pictures and sounds of the animals. Why not? It can even be done in color!

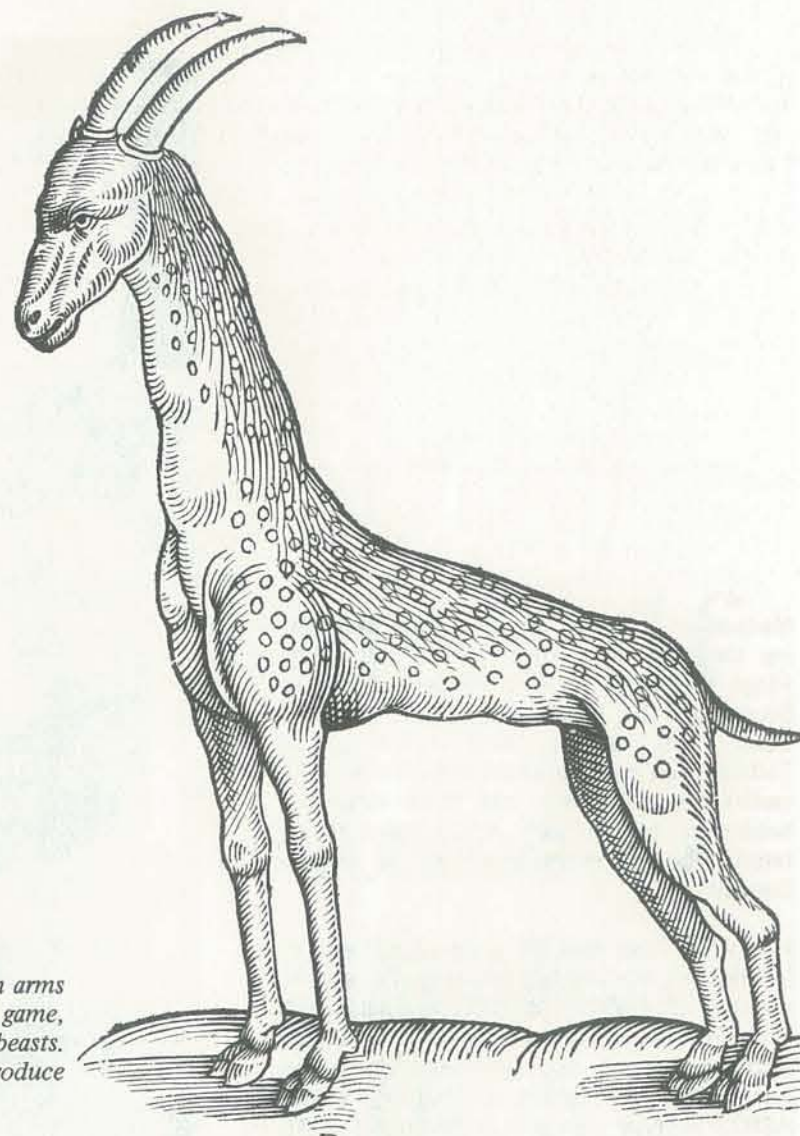
For any readers wishing to correspond with Mike or Tom, just write to M. Gabrielson, P.O.Box 2692, Stanford, CA 94305, or T. Martin, 1918 Cooley Ave., #5, East Palo Alto, CA 94303. -RZ

After repeatedly running into demonstrations of the Animal game and its variations, we finally decided to implement a version ourselves.

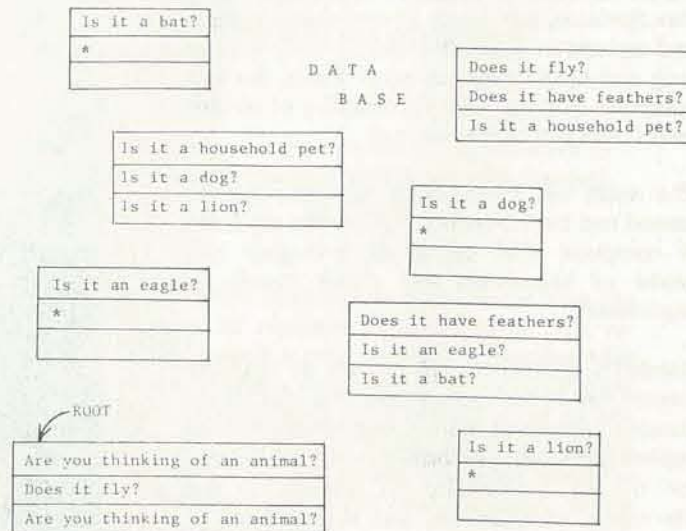
DESIGN

Animal relies on a growing data base of records. Each record consists of three parts.

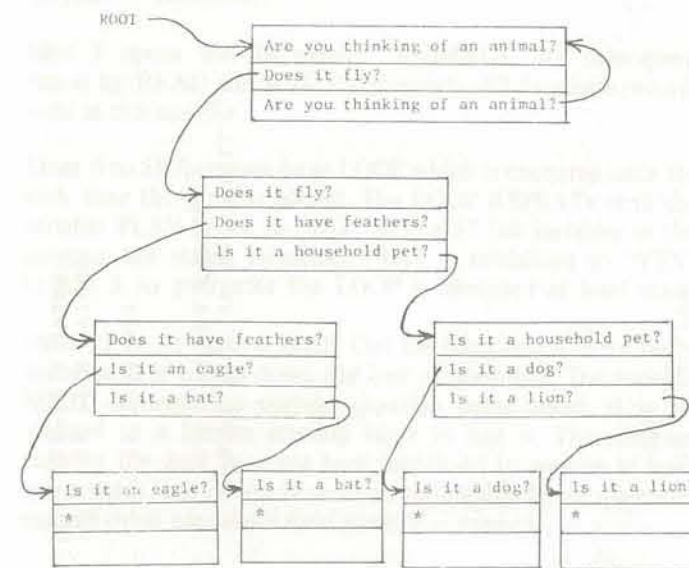
- (0) A question to ask.
- (1) The next question to ask if the answer to (0) was "yes".
- (2) The next question to ask if the answer to (0) was "no".



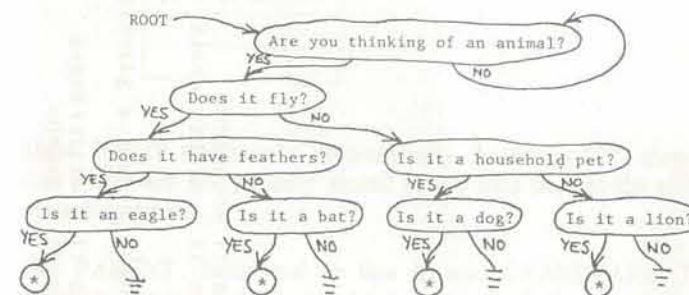
Sometimes part (0) is a final guess with no follow-up questions. In that case, part (1) is simply a "*", and part (2) is null. For example, here is a data base of eight records:



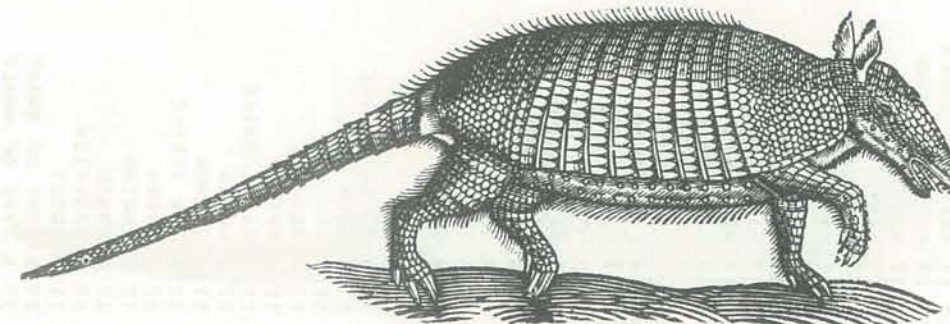
The root record is used by the computer to start the questioning. Notice how the records can be arranged into a binary tree, with a path leading from the root to every question in the data base.



Here's the same tree, shown with a little less redundancy:



To play the game, the computer starts with the root question, and then asks subsequent questions as determined by the player's yes or no answers. If an "*" is reached, the computer has "guessed" the player's animal. If a null is reached, the computer has failed, and the player is asked for a new animal to be added to the data base.



Listing 1 is a sample dialogue between the computer and a player. The resulting data base is summarized in Listing 2.

Listing 1
Sample Run

```
:ANIMAL
ARE YOU THINKING OF AN ANIMAL? (YES OR NO)?NO
ARE YOU THINKING OF AN ANIMAL? (YES OR NO)?YES
DOES IT LIVE UNDER WATER?NO
DOES IT HAVE WINGS?NO
DOES IT HAVE LEGS?YES
DOES IT HAVE FOUR LEGS?YES
DOES IT EAT MEAT?YES
IS IT A LION?NO
I GIVE UP! WHAT IS YOUR ANIMAL?A DOG
BEFORE I ASK 'IS IT A LION?',
WHAT NEW QUESTION CAN I ASK THAT WILL UNIQUELY IDENTIFY
A DOG?IS IT A COMMON HOUSE PET
WHEN THINKING OF A DOG,
WHAT IS THE ANSWER TO 'IS IT A COMMON HOUSE PET'?YES
I'LL REMEMBER THAT.
DO YOU WANT TO PLAY AGAIN?YES
ARE YOU THINKING OF AN ANIMAL? (YES OR NO)?YES
DOES IT LIVE UNDER WATER?NO
DOES IT HAVE WINGS?NO
DOES IT HAVE LEGS?YES
DOES IT HAVE FOUR LEGS?YES
DOES IT EAT MEAT?YES
IS IT A COMMON HOUSE PET?YES
IS IT A DOG?YES
I GUESSED YOUR ANIMAL!
DO YOU WANT TO PLAY AGAIN?NO
```



```

ANIMAL
001 * ANIMAL = MIKE GAHRIELSON - 3/19/78
002 *
003 OPEN "", "ANIMALS" ELSE PRINT "ANIMALS FILE IS MISSING!" ; STOP
004 *
005 PLAY = "YES"
006 LOOP UNTIL PLAY = "NO" DO
007   ANSWER = ""
008   PARENT = ""
009   ROOT = "ARE YOU THINKING OF AN ANIMAL? (YES OR NO)"
010 *
011   LOOP WHILE (ROOT <> "*" ) AND (ROOT <> "" ) DO
012     OLDANSWER = ANSWER
013     PRINT ROOT:
014     INPUT ANSWER
015     READ RECORD FROM ROOT ELSE PRINT "ANIMALS FILE IS DAMAGED!" ; STOP
016     GRANDPARENT = PARENT
017     PARENT = ROOT
018     IF ANSWER = "YES" THEN ROOT = EXTRACT(RECORD, 1, 0, 0) ELSE
019       ROOT = EXTRACT(RECORD, 2, 0, 0)
020     END
021   REPEAT
022 *
023   IF ROOT = "*" THEN PRINT "I GUESSED YOUR ANIMAL!" ELSE
024     PRINT "I GIVE UP! WHAT IS YOUR ANIMAL":
025     INPUT NEWANIMAL
026     WRITE "*" ON "IS IT ":NEWANIMAL
027     USED = "YES"
028 *
029   LOOP
030     PRINT "BEFORE I ASK '":PARENT:"?', "
031     PRINT "WHAT NEW QUESTION CAN I ASK THAT WILL UNIQUELY IDENTIFY"
032     PRINT NEWANIMAL:
033     INPUT QUESTION
034     READ RECORD FROM QUESTION ELSE USED = "NO"
035     WHILE USED = "YES" DO
036       PRINT "SORRY...I ALREADY USE THAT QUESTION."
037     REPEAT
038 *
039     PRINT "WHEN THINKING OF '":NEWANIMAL:" ", "
040     PRINT "WHAT IS THE ANSWER TO '":QUESTION:"'":
041     INPUT ANSWER
042     IF ANSWER = "YES" THEN
043       WRITE "IS IT ":NEWANIMAL:CHAR(254):PARENT ON QUESTION
044     END ELSE
045       WRITE PARENT:CHAR(254):"IS IT ":NEWANIMAL ON QUESTION
046     END
047     READ RECORD FROM GRANDPARENT ELSE PRINT "I'VE LOST MY MEMORY!" ; STOP
048     IF OLDANSWER = "YES" THEN
049       WRITE REPLACE(RECORD, 1, 0, 0, QUESTION) ON GRANDPARENT
050     END ELSE
051       WRITE REPLACE(RECORD, 2, 0, 0, QUESTION) ON GRANDPARENT
052     END
053 *
054     PRINT "I'LL REMEMBER THAT."
055     END
056   PRINT "DO YOU WANT TO PLAY AGAIN":
057   INPUT PLAY
058 REPEAT
059 *
060 END

```

Listing 3 ANIMAL PROGRAM

When the computer finally EXTRACTs an "*" or null, the WHILE condition in line 11 will no longer hold, the LOOP terminates, and line 23 is reached. If the final ROOT is "*", the computer has won and reports success, ELSE lines 24 to 55 are executed in order to learn a new animal.

The colon seen in line 24 is the concatenation operator, which often appears as a semicolon in other BASICs.

The user types in the name of the new animal when line 25 is executed. To simplify the code, the program assumes the user will provide the necessary indefinite article and avoid any terminating punctuation. In other words, the program expects input like "A CHICKEN" or "AN OSTRICH" with no ending period.

Line 26 creates a new data base record for the new animal. The "*" is part (1), part (2) is null, and the record's key in part (0) is formed by prefixing "IS IT" to the animal's name.

Whenever a new animal is added to the data base, a new question must also be added that will allow the computer to differentiate between the new animal and the last guess it made. Lines 30 to 32 ask for this somewhat awkwardly. A better prompt might be something like:

WHAT QUESTION CAN I ASK TO TELL A COW
FROM A HORSE?

but this would require temporarily stripping the "IS IT" from the PARENT. The current phrasing is used to keep the code uncluttered and hopefully more understandable. For the same reasons, the program assumes the player will type in a question with no terminating question mark or other special punctuation in response to line 33.

The LOOP in lines 29 to 37 accepts questions from the player until one is found which is not already used in the data base for distinguishing between two already-known animals. A

totally new question is needed since the program will make one of its branches point to the new animal, and existing questions already have both their branches used up.

The variable USED (initialized in line 27) controls the LOOP; its value of "YES" assumes that any question suggested by the user has already been used in the data base. Line 34 attempts a READ of each proposed new question. If the READ fails, then the question has not been used, the ELSE clause is executed to set USED to "NO", the WHILE test in line 35 will fail, and line 39 will be reached.

Lines 39 to 41 determine which branch of the new question record should point to the new animal. The actual question record is created by line 43 or 45. CHAR (254) generates the ASCII code for the special system delimiter character that separates parts (1) and (2) in data base records.

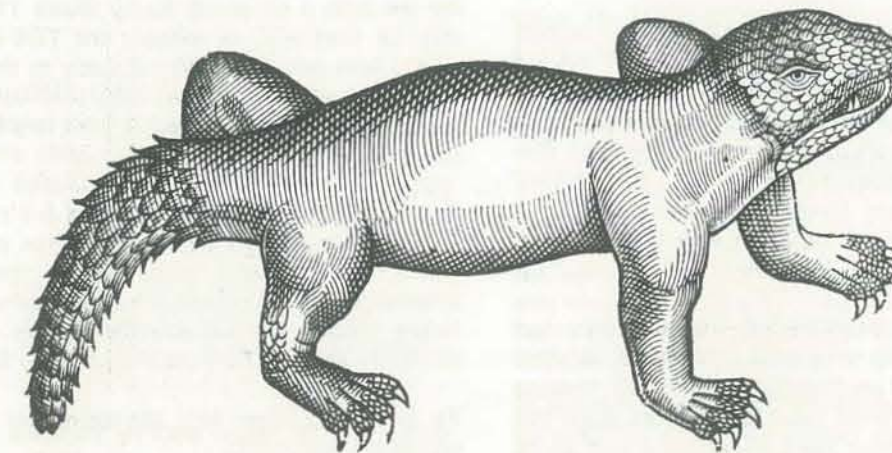
Lines 47 to 52 complete the insertion of the new question into the data base tree. The READ in line 47 fetches the record pointing to (containing the key of) the computer's last guess. The pointer (record key) is then REPLACed with a pointer to (the key of) the new question. The patched GRANDPARENT record is then written back out to disk to complete the operation of inserting the new question.

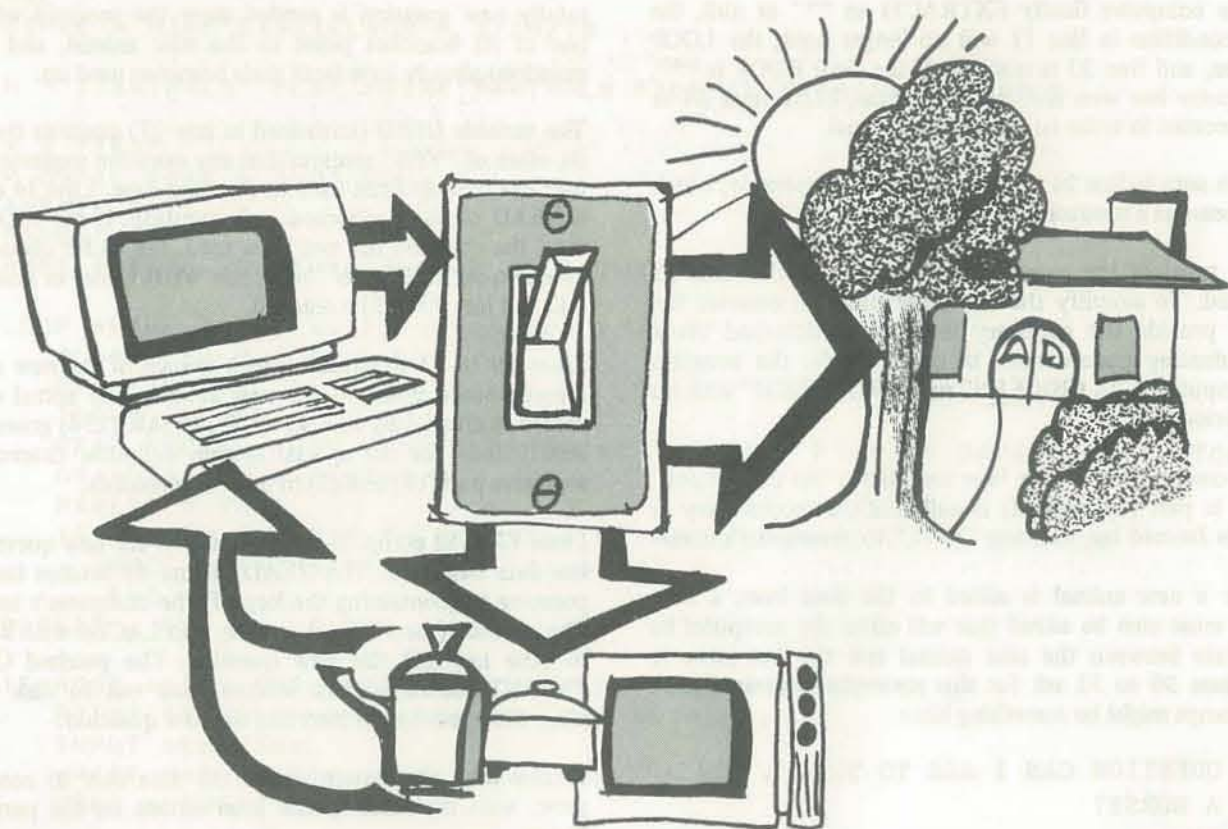
In this way, the system allows the data base to continually grow, with no other special intervention on the part of the player.

FINAL NOTES

For a description of implementing binary trees in assembly language, see "Binary Tree Manipulation on the 8080," *Dr. Dobb's Journal*, # 30, page 15.

Animal is a fascinating game to play on a computer, since the machine participates in such a lively way. We hope this article will stimulate new versions and implementations of Animal.





The Outside Connection

BY DOC PLUMBER

Controlling a home alarm system with your TRS-80? Switching on the coffee maker with your PET? Yes, you can connect your micro to the "outside world," and you don't have to be a super programmer or engineer to make the modifications. According to writer Doc Plumber, any amateur can! Doc defines an amateur as "someone who knows a little BASIC and can tell the difference between a screwdriver and a wrench." (Plumber is otherwise known as Don Inman, former editor of Calculators/Computers Magazine.) —LB

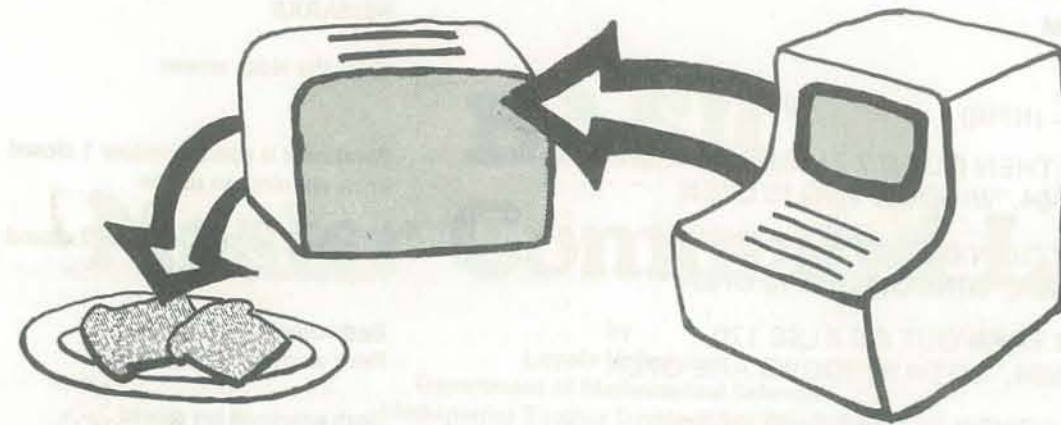
This is the first in a series of articles on how an amateur can experiment with connecting a computer to outside devices. If you've never attempted an "outside connection" because you thought it demanded great technical skill, we hope this article will dispel some of your fears. In fact, this first device has built-in input/output ports, making hardware experience unnecessary.

The VAR/80, manufactured by Telesis Laboratory, is designed for use with a minimum Radio Shack TRS-80 computer. It may be used with or without the TRS-80 expansion interface. I have connected mine directly to the connector on the TRS-80 keyboard module. The unit comes assembled and tested, and a 40-pin connector with two feet of ribbon cable is supplied.

The VAR/80 provides you with one 8-bit input port and one 8-bit output port. A screwdriver is the only tool needed to connect your inputs or outputs to the unit. Screw-type terminal strips are provided for easy connection. In this and future articles, we will describe a series of simple examples of INPUT and OUTPUT statements using the VAR/80.

To get information into the computer from the outside, we use the statement:

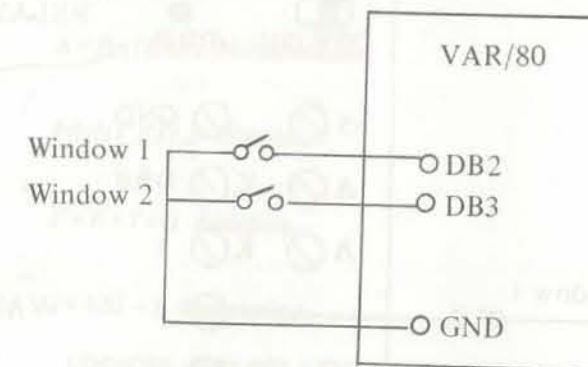
INP (0)



SIMPLE, SAMPLE PROGRAM

(This is adapted from a program in the VAR/80 manual.)

Suppose you have two windows in your house wired to two inputs of the VAR/80 so that you will be able to determine whether the windows are open or closed. This could be done with two pieces of conducting material which make contact when the windows are closed, but do not make contact when the windows are open. In our example, we will make the connections to input bits 2 and 3 of the VAR/80 from one contact at each window. The other contacts from each window are connected to ground as shown below.



When the VAR/80 is being used, all inputs are normally at a high level value (equal to one). Therefore, when the two windows are open, the input value is: $128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$. If we make our input statement read:

$$X = 255 - \text{INP } (0)$$

we have the following values for X:

only window 1 closed	$X = 255 - (128+64+32+16+8+0+2+1) = 4$
only window 2 closed	$X = 255 - (128+64+32+16+0+4+2+1) = 8$
both windows closed	$X = 255 - (128+64+32+16+0+0+2+1) = 12$
both windows open	$X = 255 - (128+64+32+16+8+4+2+1) = 0$

In our program, we will test the value of X to see if windows are open or closed. We will then use output bits DB0 and DB1 to turn on lights to show which windows are open. We could also wire up these bits to an alarm system.

The VAR/80 is wired so that it uses port number zero. Since there may be more than one port in use, each must be designated by the appropriate number. The port number must be enclosed in parentheses.

To send information to the VAR/80 from the computer, we use the statement:

OUT 0,64

No parentheses this time. The number to the left of the comma is the port number. The number on the right of the comma tells which bits of the 8-bit output should be equal to one (all others will be zero).

Remember that the computer is working with binary numbers. Since BASIC works in decimal numbers, each bit of the 8-bit input or output is assigned a weighted value. These values are shown below:

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
128	64	32	16	8	4	2	1

DB for data bit

Thus the statement: OUT 0,64 would output a 1 (voltage level high) to the output port. All other outputs would be at a low level (near zero volts, which is called logic level zero). In other words, the computer is really outputting the binary equivalent:

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	1	0	0	0	0	0	0	
							0 = 64	
								(decimal)

At the output port, the data bit DB6 would have a voltage of +2.4 to +5 volts and all others would be near zero. The voltage at bit 6 could be used to control some external device.

THE PROGRAM

```

90 CLS
100 X = 255 - INP(0)

110 IF X = 4 THEN OUT 0,2 ELSE 130
120 PRINT @64,"WINDOW TWO IS OPEN "

130 IF X = 8 THEN OUT 0,1 ELSE 150
140 PRINT @64,"WINDOW ONE IS OPEN "

150 IF X = 0 THEN OUT 0,3 ELSE 170
160 PRINT @64,"BOTH WINDOWS ARE OPEN"

170 IF X = 12 THEN OUT 0,0 ELSE 100
180 PRINT @64,"BOTH WINDOWS CLOSED "

190 GOTO 100
    
```

REMARKS

Clear the video screen

Window 2 is open, window 1 closed
Print warning on screen

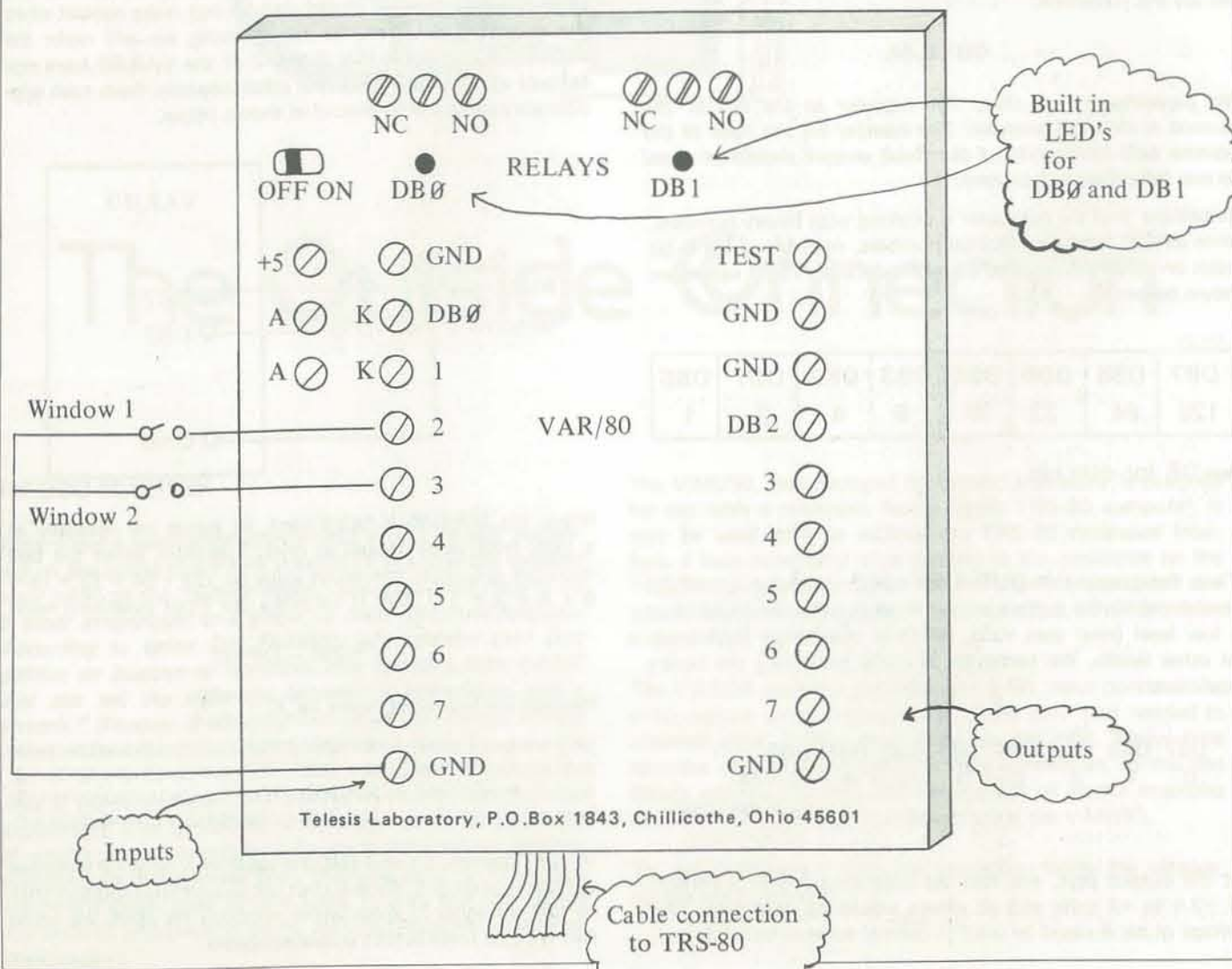
Window 1 is open, window 2 closed
Print warning on screen

Both windows are open
Print warning on screen

Both windows are closed
Print message

Go back and check again

VAR/80 CONNECTIONS



BASIC Master Command List

by
 Loyola University
 Department of Mathematical Sciences
 Mathematics Teacher Development Program
 Program Director: Dr. R. T. McLean
 Instructors: Dr. Antonio M. Lopez, Jr., Dr. James Schmit
 Student Assistant: Mr. David Griffen
 (In conjunction with the National Science Foundation)

What? Write programs that run on the SOL-20, PET 2001, Apple II and Level II TRS-80 machines without changes? Unheard of! There are more than three statements that work the same on all machines? Sounds like too much crayfish gumbo to me!

"If teachers stick to the list, they can exchange listings and in effect exchange programs," notes Dr. Lopez.

Then Dr. Jane Donnelly Gawronski of the Department of Education, San Diego, CA, puts me in touch with Dr. Lopez and the Master Command List and dispels my skepticism.

This rather large subset of BASIC seems to make such exchanges easy and possible. The list was used and tested in an NSF-sponsored program at Loyola University. We and Dr. Lopez would like to hear from people who use the list to develop "machine" independent programs. — RZ

COMMANDS

LIST
 LIST X Lists line "X" if there is one.
 LIST Lists the entire program.
 LIST X-Y Lists all the lines within a program from X to Y.

RUN
 RUN Starts execution of the program in memory at the lowest numbered statement. RUN initializes all variables to zero.
 RUN 200 Starts run at the specified line.

NEW
 NEW Deletes current program and clears all variables.

ARITHMETIC OPERATORS

=
 A = 100 Assigns a value to a variable.

-
 B = -A Negation.

^
 PRINT X^3 Exponentiation (equal to X*X*X in the sample statement). A^B with A negative and B not an integer gives an error.

*
 X = R*(B*D) Multiplication.

/
 PRINT X/1.3 Division.

+
 Z = R+T+Q Addition.

-
 J = 100 - I Subtraction.

LOGICAL AND RELATIONAL OPERATORS

=
 IF A = 15 THEN 40 Equals.

<>
 IF A <> 0 THEN 5 Not Equal.

>
 IF B > 100 THEN 8 Greater Than.

<
 IF B < 2 THEN 10 Less Than.

<=
 IF 100 <= B+A THEN 1 Less Than or Equal.

>=
 IF Q >= R THEN 20 Greater Than or Equal.

★ References: User manuals for the Level II TRS-80, PET 2001, Apple II and SOL-20.

AND
IF A < 2 AND B < 1 THEN 5 If expression (A < 2) AND expression (B < 1) are both true, branch to line five.
IF A < 1 OR B < 2 THEN 2 If either expression (A < 1) OR expression (B < 2) is true, then branch to line two.

STATEMENTS

DIM
DIM A(3), DIM R3(5,5) Allocates space for matrices. All matrix elements are set to zero. Matrices can have up to 255 dimensions. The size of each must be less than 32767, and is limited by total memory available.

END
END Terminates program execution. End can be used anywhere in the program.

FOR
FOR V=1 TO 9.3 STEP 6 The initial statement of a FOR-NEXT loop. V is set equal to the value of the expression following the equal sign. This value is called the initial value. Then statements between FOR and NEXT execute. The final value is the value of the expression after the TO. The step is the value of the expression following STEP. When the NEXT statement is encountered, the STEP is added to the variable. If no STEP size is given, it is assumed to be one (1).

GOSUB
GOSUB 910 Branches to the specified statement (910). The program then starts executing at line 910 until a RETURN is encountered; where a branch is then made to the statement after the GOSUB. GOSUB nesting is limited only by available-memory.

IF... THEN
IF X < 0 THEN 5 If the relation is true, **IF X < 0 THEN PRINT "X"** branches to the specified statement or executes the statements on the line after the THEN.

INPUT
INPUT V,W,W2 Requests data from the keyboard (to be typed in). Each value must be separated from the preceding value by a comma (,). The last value must be followed by a carriage return. A "?" is typed as a prompt character.
INPUT "VALUE";V Optionally, may be used to type a message before requesting data from the terminal.

NEXT
NEXT V Marks the end of a FOR-NEXT loop.

PRINT
PRINT X,Y,Z Prints the value of expressions in the list. If the value list to be printed out does not end with a comma (,) or a semicolon (;), then a carriage return/line feed is executed after all the values have printed. Strings in quotes (") may also be printed. If a comma appears after an expression in the list, then spaces are output until the beginning of the next pre-set column field is reached.
PRINT X,Y
PRINT "VALUE IS";A

RETURN
RETURN Causes a subroutine to return to the next statement after the most recently executed GOSUB.

INTRINSIC FUNCTIONS

ABS (X)
PRINT ABS (X) Gives the absolute value of the expression X. ABS returns X if X >= 0, -X otherwise.

ATN (X)
PRINT ATN (X) Gives the arctangent of the argument X. The result is returned in radians and ranges from -π/2 to π/2. (π/2 = 1.5708)

COS (X)
PRINT COS (X) Gives the cosine of the expression X. X is interpreted as being in radians.

EXP (X)
PRINT EXP (X) Gives the constant e=2.71828 raised to the power X (E^X).

INT (X)
PRINT INT (X) Returns the largest integer less than or equal to the argument X. For example: INT (.23) = 0; INT (7) = 7

LOG (X)
PRINT LOG (X) Gives the natural (base e) logarithm of its argument X.

RND (X)
PRINT RND (X) Generates a random number between 0 and 1 for X = 0.

SIN (X)
PRINT SIN (X) Gives the sine of the expression X. X is interpreted as being in radians.

SQR (X)
PRINT SQR (X) Gives the square root of the argument X. An error will occur if X is less than zero.

TAB (X)
PRINT TAB (X) Spaces to the specified position on the screen. May be used only in PRINT statements. It specifies the absolute position from the left-hand margin where printing is to start.

TAN (X)
PRINT TAN (X) Gives the tangent of the expression X. X is interpreted as being in radians.

STRING FUNCTIONS

LEFT\$ (X\$,I)
PRINT LEFT\$ (X\$,I) Gives the leftmost I characters of the string expression X\$. If I <= 0 or > 255 an error occurs.

LEN (X\$)
PRINT LEN (X\$) Gives the leftmost I characters of the string expression X\$ in characters (bytes). Non-printing characters and blanks are counted as part of the length.

MID\$ (X\$, I, J)
PRINT MID\$ (X\$,I,J) MID\$ is called with 3 arguments and returns a string expression composed of characters of the string expression X\$ starting at the I'th character, for J characters. If J specifies more characters than are left in the string, then all characters from the I'th on are returned.

RIGHT (X\$, I)
PRINT RIGHT\$ (X\$, I) Gives the rightmost I characters of the string expression X\$. If I <= 0 or > 255 an error will occur. If I >= LEN (X\$), then RIGHT\$ returns all of X\$.

STR\$ (X)
PRINT STR\$ (X) Gives a string which is the character representation of the numeric expression X.

VAL (X\$)
PRINT VAL (X\$) Returns the string expression X\$ converted to a number.

A string may be from 0 to 255 characters in length. All string variables end in a dollar sign (\$); for example: A\$, B9\$, K\$.

The symbol "+" is used to join or concatenate strings. In the statement:

$$Z\$ = R\$ + Q\$$$

Z\$ becomes R\$ with Q\$ coming right after it. The resulting string must be less than 255 characters in length or a "STRING TOO LONG" error will occur.

Spanish BASIC

BY JIM DAY

Jim sent us this note and we thought it made a good supplement to the BASIC Master Command List article on the preceding pages. FIN.

- RZ

Why should Spanish-speaking BASIC programmers have to put up with English mnemonics? Why not give them the option of using Spanish mnemonics instead? The same interpreter could handle both forms and the changes would be trivial. I suggest the equivalents shown on the enclosed table.

DIRECT COMMANDS

DEL	TACHA
LIST	LISTA
LOAD	PONE
RUN	ANDA
SAVE	SALVA

INDIRECT COMMANDS

DATA	DATOS
DEF FN	DEF FN
DIM	DIM
END	FIN
FOR...TO...STEP	DE...A...GRADA
GOSUB	VASUB
GOTO	VA A
IF... THEN	SI... LUEGO
INPUT	ENTRA
LET	HACE
NEXT	PROXIMO
ON...GOTO	POR... VA A
PRINT	TIPO
READ	LEE
REM	NOTA
RESTORE	RESTAURA
RETURN	RETORNO

Commodore's PET is a factory-assembled personal computer based on a 6502 microprocessor. The original PET, model 2001-8, is a \$795 system that includes a keyboard, cassette tape unit, built-in TV screen, some graphics, upper and lower case, extended 8K BASIC, and 8K of user memory.

SPOT is devoted to the host of applications—routine and wild—which PET users have found for their machines, as well as to the nitty-gritty of repairs and modifications. In other words, almost anything relating to the PET is fit material for this column. Just send Harry your questions, ideas, and tapes c/o PCC. He'll give each of them his careful attention. —LB

GAMES, THESE ARE ONLY GAMES

This month's column is devoted to a subject that occupies a lot of PET hours, though many users don't want to admit it. Yes, I mean games. But these games tax the mind; they're a far cry from the simulations of video games which were a big hit in the early days of the PET and other personal computers. These are games that the whole family can get involved in, that teach problem solving and planning or such specific skills as map-making and exploring. They are games that you build yourself with a set of tools provided by the program creator. Games you can easily spend weeks with. You're not just hitting a moving target on the screen.

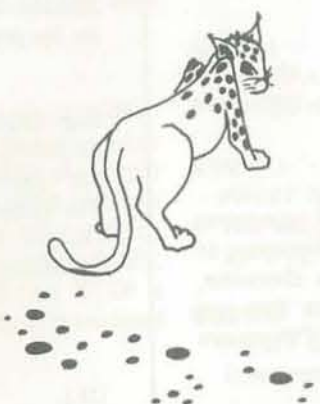
I think these programs represent a turning point for microcomputing. They are truly sophisticated. Even if you're an expert programmer, you can't just take the idea, and throw together your own version over a weekend. These games took the authors hundreds of hours of work, not just programming, but researching background information for the scenarios.

Of course, they are still games. But they show there are people out there willing to invest the time and energy to do first-rate complex programs. I assume we'll soon see the same amount of talent going into business applications; to date most business-oriented software is thrown together, and often better rewritten from scratch rather than purchased and modified. I have seen a few packages under development at Commodore that look like they'll be first-rate large packages when completed. Get ready for them!

SPOT

The Society of
PET Owners and Trainers

BY HARRY SAAL



TANKTICS AND LEGIONNAIRE

These two games are written by Chris Crawford, and are available from the Pleiades Game Co., 202 Faro Ave., Davis, CA 95616. Each is run on the "traditional" hex grid of the war gamer, but that's where the similarity ends. *Tanktics* is very cerebral; those of you familiar with war simulation strategy games will feel at home here. It comes with a map of tank terrain over which you maneuver your forces. The PET serves as both your opponent, maneuvering his forces according to the same rules, and also as your battle computer, keeping track of your moves and feeding you information about what you can see out your tank turrets. It's an interesting and complex game, and seems ideal for war game freaks who don't have a partner 24 hours a day.

Legionnaire is more straightforward. You see a grid on the PET screen and have to hunt the enemy by moving around; at the same time he is hunting you. This all happens in real time, and you are constantly feeding separate orders to each of your "tribes," using the PET keyboard. People not into war games will probably do well to start with this one. It is fast moving, easy to learn, and takes quite a bit of planning to do well. *Legionnaire* costs \$9 and *Tanktics* \$15 (plus 6% tax for California residents).

MAY 1941

This excellent game is an historical simulation of the search for the German battleship *Bismarck* by the British during May 1941. It is related in complexity and realism to the *Tanktics* game above, but has a significant difference. It is not just a physical simulation of the battle, but attempts to be a realistic simulation of actual events. The manual which comes with *May 1941* is a gem, containing far more than playing instructions. It describes the historical event being simulated, the nature and limitations of the ships and planes involved (for example, the probability of spotting the *Bismarck* decreases at night, and during the day of May 23, when there was bad weather in the Atlantic), and the general strategy to use. The biggest problem is locating the *Bismarck* and having forces ready to then chase and engage in battle. It takes patience and planning just to accomplish that much.

The method of entering moves is simple, and the display contains a nice map of the North Atlantic on the righthand side, showing the general land masses and position of ships as they are moved about. History buffs and war gamers will find this game a delight. It is available from Alderaan, P. O. Box 1243, So. Pasadena, CA 91030 for \$24.95 (plus 6% tax for California residents).

STARFLEET ORION

Starfleet Orion is an interesting concept; unfortunately, it takes little advantage of some of the graphic capabilities of the PET and its excellent video display. Basically, *Starfleet Orion* is a two-player game between opposing forces. Each has ships of differing capabilities, and can enter a series of orders for each without the opposing player observing them. Once done, the two forces engage in battle, and this continues until one of the two sides

is destroyed. Ships can move, attack with torpedoes or beams, use shields for protection, etc. It is a shame that the human factors and the computer's capability in move determination aren't used better because the basic premise is pretty interesting. The Battle Manual that comes with *Starfleet Orion* gives 12 different scenarios which can be entered, each producing very different challenges. *Starfleet Orion* is available for \$16.95 from Automated Simulations, P. O. Box 4232, Mountain View, CA 94040.

QUEST

Quest is a challenging diversion in which you must locate a treasure and then escape from a series of interconnected rooms within a 3-dimensional maze. At each turn you can try to move North, East, South, West, Up or Down as you attempt to avoid various traps and problems. It is an engrossing activity, and is especially educational for young children learning about maps and navigation. The descriptions are clever, and the interconnections between rooms not simple at all. For example, one of the more challenging paths you must traverse takes you through rooms called "a little twisty maze," "a little twisting maze," "a twisty little maze," or "a twisting little maze!" *Quest* is available from the Peninsula Computer Project, Peninsula Way, Menlo Park, CA 94025 for \$9.95 (plus 6% tax for California residents).

HUNT AND HUNTWRITER

The May-June issue of *Recreational Computing* contained an article by Michael Richter describing his meta-games: *Hunt* and its game generator program, *Huntwriter*. *Hunt* combines the environment of *Quest* with the flexibility of *Starfleet Orion* or *Adventure* (below). *Hunt* is a table-driven game in which the scenarios (generally fantasy simulations) are provided in the form of table or data tapes, which are first read in. The neat thing about *Hunt* is that you can write your own game using *Huntwriter*. While it is limited compared to *Quest* or *Adventure*, it does allow you to build a game around a special interest (say *Oz* or a particular TV series). You can incorporate names and places you choose instead of being restricted to the author's choice. *Hunt* is now available from Computer Way, P. O. Box 7006, Madison, WI 53707 for \$10.

ADVENTURE

And now, the real thing! All these computerized fantasy simulations trace their origins back to the original granddaddy *Adventure*, written first at MIT by Willy Crowther, and then expanded extensively by Don Woods of the Stanford Artificial Intelligence Laboratory. Those games are based on the non-computerized fantasy game, *Dungeons and Dragons*, by Dave Arneson and Gary Gygax. The computer versions (see *RC*, July-August 1979, for an article on *Zork*) are all large, complex programs, which occupy hundreds of kilobytes of storage, and typically take tens to hundreds of hours to play.

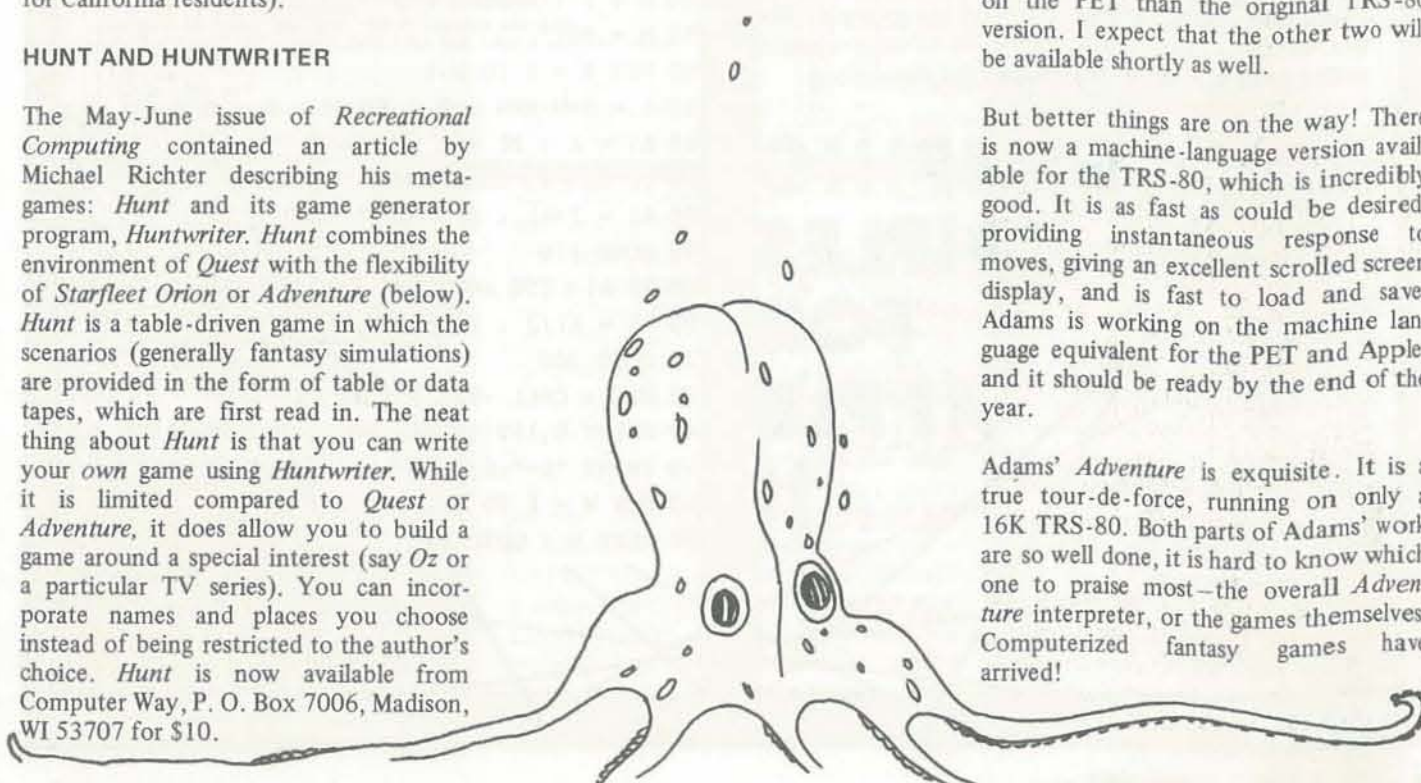
They contain a delightful fantasy atmosphere, with literally dozens of puzzles that need to be solved in your quest for treasure and points. Numerous challenges need to be met, using objects discovered along the way, moving them from place to place, with nary a clue from the program about what is going on. You learn as you go, and have to be very creative to survive more than a few moves. In fact, these games are somewhat like intricate Chinese wooden puzzles—there is really only one way to do them; any little mistake along the way, and you are hopelessly lost, and must begin again.

Adventure, by Scott Adams, is the microcomputer realization of these big sophisticated mainframe games. Adams' version is also table-driven, and the overall structure is described in an article which appeared in the *NCC '79 Personal Computing Proceedings*, pp. 218-223, entitled "An Adventure in Small Computer Game Simulation." (It was scheduled to be reprinted in the August 1979 issue of *Creative Computing*.) *Adventure* provides a far richer environment for constructing games than Richter's *Huntwriter*; of course, this means that one can't easily generate one's own, and unfortunately Adams doesn't (yet) distribute his meta-program for game-building and editing. There are already three different adventure games running with the Adams interpreter; each takes literally hundreds and hundreds of moves to complete (fortunately, you can save a game in progress) and have taken hundreds and hundreds of hours to produce.

The first version I saw was a cassette-based version for the TRS-80. It was written entirely in BASIC, and was excruciatingly slow. It has been converted for the PET, and is available from AB Computers, 115 E. Stump Road, Montgomeryville, PA 18936, for \$7.95. Note that it runs *only* on PETs which are 24K or bigger. This is the first of the three *Adventures*, and is quite a bit faster on the PET than the original TRS-80 version. I expect that the other two will be available shortly as well.

But better things are on the way! There is now a machine-language version available for the TRS-80, which is incredibly good. It is as fast as could be desired, providing instantaneous response to moves, giving an excellent scrolled screen display, and is fast to load and save. Adams is working on the machine language equivalent for the PET and Apple, and it should be ready by the end of the year.

Adams' *Adventure* is exquisite. It is a true tour-de-force, running on only a 16K TRS-80. Both parts of Adams' work are so well done, it is hard to know which one to praise most—the overall *Adventure* interpreter, or the games themselves. Computerized fantasy games have arrived!



FuturePlay™


(A LOOK AT FUTURE PCC & RC FANTASIES)

Music in the Park

Kepler's Bookstore

Computers in the Library

Canada College



Have you heard of ComputerTown, U.S.A.!?
Yes! No?

Peninsula School

Menlo-Atherton High School

Send your ideas/inquiries to:
ComputerTown, U.S.A.!
P.O. Box 310
Menlo Park, CA 94025
Enclose a S.A.S.E. if you want information

Round Table Pizza

BY JIM DAY

The Applesoft program shown below generates and displays Pythagorean triples. The dimensions of a right triangle having integral sides are determined by the relationships defined in line 150 of the program. The rest of the program draws a picture of each triangle, scaled to fit the screen, and displays the dimensions.

GRAPHIC TRIPLES FOR APPLE II

```

100 REM PYTHAGOREAN TRIPLES
110 CALL -936
120 M = 1 : HCOLOR = 3
130 M = M+1
140 FOR N = 1 TO M-1
150 A = M*M-N*N : B = 2*M*N : C = M*M+N*N
160 A1 = A : B1 = B : C1 = C
170 IF C1 > 75 THEN 200
180 A1 = 2*A1 : B1 = 2*B1 : C1 = 2*C1
190 GOTO 170
200 IF A1 < 150 AND B1 < 150 THEN 230
210 A1 = A1/2 : B1 = B1/2 : C1 = C1/2
220 GOTO 200
230 HGR : CALL -936 : VTAB 22
240 HPLLOT 0,150 TO B1,150 TO B1,150-A1 TO 0,150
250 PRINT "A=";A;" B=";B;" C=";C
260 FOR W = 1 TO 3000 : NEXT W
270 NEXT N : GOTO 130
  
```

LITTLE PROGRAMS FOR GIANT COMPUTERS

APL

BY NICK COOLEY

Nick sends these two APL procedures to demonstrate the "power and beauty" of the language. Nick is with I. P. Sharp Associates, Inc., in Chicago. The "processing heart" of the Sharp network is an Amdahl 470 V6-II and an IBM 360/75. The first function, MTMIND, is the codebreaker; MTMIND2 the codemaker. The challenge? Are there shorter BASIC versions?

-RZ

APL

MTMIND

```

VMTMIND[ ]V
V MTMIND:CNTR:CDST:TRYNOC
[1] MOKT A LIST OF ALL POSSIBLE CODES MADE FROM THE NUMBERS
    1 2 3 4 5 6. DUPLICATES ARE ALLOWED.
[2] COS=(1000*16)+*(100*16)+*(10*16)+*16
[3] CNT=0
[4] L1:'TRIAL ':(CNT-CNT+1
[5] MCOMPUTE AND OUTPUT GUESS
[6] 'I GUESS ':TRY+. 10 10 10 10 TCDST[?DCDS]
[7] MINPUT THE RESPONSE
[8] L2:GOS=7,0pT+ENTER BLACKS AND WHITES:'
[9] MCHECK TO SEE THAT THE INPUT IS PLAUSIBLE.
[10] *(4*+/GDS)Y(A/GDS= 3 1)Y2#GDS)/'-L2,0pT+'IMPOSSIBLE
    COMBINATION''
[11] MCHECK TO SEE IF GAME IS OVER.
[12] -(#GDS[1])pL3
[13] MGENERATE ALL CODES THAT DO NOT FIT THE INFORMATION.
[14] ' REMAINING NUMBER OF CODES:'(NOC-pGDS+(L,TRY+,*
    10 10 10 10 TCDST*(L+GDS)*(+/GDS)*+110 10
    10 10 TCDST*TRY))GDS
[15] MIF NO CODES ARE LEFT, PRINT MESSAGE AND QUIT.
[16] *(0=NOC)/'+*1.(0pT).0pT+'DID YOU MAKE A MISTAKE SOMEW
    HERE?''
[17] +*1
[18] L3:'GOT IT'
V
  
```

MTMIND2

```

VMTMIND2[ ]V
V MTMIND2:GOS:CNTR:TCO:CD:TOS:JOS:RKS:J:LIM:YV
[1] MGENERATE THE CODE.
[2] PLAY:CD+7 6 6 6 6
[3] CNT=0
[4] T+I HAVE MY CODE. ENTER 0 0 0 0 IF YOU GIVE UP'
[5] L1:GOS=7,0pT+ENTER YOUR GUESS'
[6] MCHECK TO SEE IF PLAYER HAS GIVEN UP.
[7] +(A/GOS= 0 0 0 0)pL5
[8] CNT+CNTR+1
[9] MCHECK FOR A WIN AND CALCULATE THE NUMBER OF BLACKS.
[10] +(RKS+X-CD-GOS)*4pL4
[11] TOS+(-X)/GOS
[12] JOS+1=0
[13] L1+TCO+(-X)/CD
[14] MCALCULATE THE NUMBER OF WHITES.
[15] L2:-L1*(1+1)pL3
[16] YV+YR+(-L1+TOS)*TCO
[17] JOS+JOS+YV
[18] TOS+1+TOS
[19] TCO+(-R)/TCO
[20] +*2
[21] MOUTPUT THE NUMBER OF BLACKS AND WHITES.
[22] L3:RKS:' BLACKS AND 'JOS:' WHITES'
[23] +*1
[24] L4:'YOU WIN IN ':(CNT)' TRIES'
[25] -(Y=1+7,0pT+PLAY AGAIN?')pPLAY
[26] +0
[27] L5:'MY CODE WAS ':(CD
V
  
```

MTMIND

```

VMTMIND
TRIAL 1
I GUESS 4 2 6 5
ENTER BLACKS AND WHITES:
?:
  2 0
REMAINING NUMBER OF CODES:150
TRIAL 2
I GUESS 4 1 5 2
ENTER BLACKS AND WHITES:
?:
  0 2
REMAINING NUMBER OF CODES:21
TRIAL 3
I GUESS 1 2 1 5
ENTER BLACKS AND WHITES:
?:
  3 0
REMAINING NUMBER OF CODES:8
TRIAL 4
I GUESS 2 2 1 5
ENTER BLACKS AND WHITES:
?:
  4 0
GOT IT
  
```

MTMIND2

```

I HAVE MY CODE. ENTER 0 0 0 0 IF YOU GIVE UP
ENTER YOUR GUESS
?:
  1 2 3 4
0 BLACKS AND 3 WHITES
ENTER YOUR GUESS
?:
  2 3 4 5
0 BLACKS AND 3 WHITES
ENTER YOUR GUESS
?:
  3 4 2 6
1 BLACKS AND 3 WHITES
ENTER YOUR GUESS
?:
  6 4 2 3
2 BLACKS AND 2 WHITES
ENTER YOUR GUESS
?:
  4 6 2 3
YOU WIN IN 5 TRIES
PLAY AGAIN?
  
```

SEPT-OCT 1979 57

Reviews

CLUSTER/ONE DISK SHARING SYSTEM

Nestar Systems, Inc.
430 Sherman Ave.
Palo Alto, CA 94306

Most teachers who have switched from time-sharing to microcomputers are delighted with the portability, the graphics possibilities, and the price. We don't miss telephone problems or down-time at all. Ad libbing a lesson when the computer was down — and chasing the problem from modem to phone line to teletype to computer and back — prematurely aged many of us.

But there are still times when it would be nice to be back in the old days: to have instant access to essentially unlimited storage, demonstration programs available to each port simultaneously, and programs secure from adolescent pranks; as well as ready access to many large programs.

Cluster/One from Nestar Systems seems to offer us the best of both worlds. Cluster/One is a disk sharing system in which up to 30 micros can share access to dual eight-inch floppies. Once a program is loaded from the disk, the micro is independent and acts as if it were alone in the room.

We emphasize to cassette users that this means instant, accurate loading and saving. For disk users, there are two important points. First, this is a very sophisticated data transmission system which uses "packet switching," a technique developed for large computer networks. The disk operating system incorporates its own error recovery: each WRITE is tried up to four times (and always verified) and each READ up to eight times. Such reliability is not currently available on mini-disk systems. Second, no individual disk drives are required, so the cost of adding additional micros excludes the disk (and typically 16K extra RAM per station for the DOS).

Right now TRS-80s, Apples, and PETs can share access to Cluster/One, all at the same time, each loading only programs written in its own version of BASIC. The system is designed for the classroom, teaching lab, or industrial environment, which requires frequent access, from many ports, to a common bank of programs secure from the casual user. We have each spent time playing with the system and are most impressed with it, both as a technical achievement and as an idea whose time is now.

Because we are educators, we think about the teaching possibilities. In a programming class, Cluster/One would eliminate the disk passing among students and maintenance of back-up copies, since the disks are centrally located. Further, all demonstration programs and common data files need only be saved onto one disk. (Consider what is involved in making even minor changes in a program located on many disks or cassettes.)

A meaningful computer assisted instruction (CAI) program using micros seems possible with this system, because each student can have access to the large number of often lengthy programs required by a sophisticated attempt at CAI. Also, all student records are maintained centrally; a student need not find the right disk before each session, and class statistics may be readily compiled.

Others will think of more applications: in business for data entry and retrieval; in computer stores for quick demonstration and comparison of different micros; or in public computer centers.

SOME DETAILS

The Cluster/One system was designed by Dr. Harry Saal, president of Nestar Systems, who is best known to RC readers for his "SPOT" column. The Cluster/One system requires the use of one 8K PET as the Cluster/One console, making the PET unavailable for other purposes. A double-sided disk system offers about 1.2 megabytes of storage (half that for single-sided). Nestar uses Shugart drives for single-sided disks and Remex drives for double.

No routine maintenance is required, although anyone purchasing a computer system should have a clear maintenance agreement with the seller. The purchase price of the entire system depends, of course, on the choice of micros, but for around \$20,000, one can be disk-sharing with 8 to 15 stations, including the price of the microcomputers.

Other computers may soon be able to be used with the Cluster/One. The dedicated PET console is slated to be eliminated, in favor of a simpler and cheaper terminal, or built-in display. Access to data files, both sequential and random, which can be written by, say, an Apple and read by a TRS-80 is planned for this year. A shared printer is expected next year, as is a hard disk, with from 10 to 100 megabytes of storage.



CRITICISMS AND COMMENTS

There is no perfect computer system, and we have a few minor criticisms. To initiate contact with the Cluster/One system, a one-line command must be typed at each station, and we would prefer doing that centrally.

We also feel that many educational users, particularly CAI, could make good use of a clock readable from a BASIC program. This is an expensive option on the Apple and the PET, and we had hoped that it could have been provided centrally by Cluster/One. Ideally, the clock should time from fractions of a second to a year and be backed up by an inexpensive battery which needs no recharge and lasts a year.

Further, not all dialects of BASIC are supported by the Cluster/One. Programs in Apple Integer BASIC and in Radio Shack Level I cannot be read from the disk. It should be noted, however, that Apple now offers Applesoft, its floating point BASIC, which is supported by Cluster/One, in ROM for the same price now charged for Integer BASIC in ROM — and that Integer BASIC is essentially a subset of Applesoft.

Another possible drawback is that the computers must be centrally located, since the maximum distance of any micro

from the disk system is 250 feet (although Nestar has plans for extending this range). If a remote demonstration, development, or use of part of the system is anticipated, an extra mini-disk drive may be needed for each type of micro.

A real plus is that this system puts a student directly in touch with the components of a miniature computer system. Under timesharing, students saw only terminals and had to trust that indeed somewhere there was a computer. The Cluster/One concept is a compromise between these extremes of involvement. Teachers will differ about the choices, but we, at least, think that disk sharing offers the right mix of security and direct access.

SUMMING UP

A Cluster/One system may be seen at the Lawrence Hall of Science in Berkeley, where it is in daily use as part of the Hall's program of providing public access to computers. We think this system is an important addition to the world of educational computing, and worth careful consideration by those about to embark on an extensive teaching program using computers.

Finally, we would like to add that anyone facing a decision to purchase a system of this magnitude will have a very difficult task sorting out all the claims and counter-claims. We urge you to seek help from a local group of educational computer users, such as the Minnesota Educational Computer Consortium, Oregon Council of Computer Educators, or a new group in northern California, Computer Using Educators. Sandy Wagner can supply information about groups in other areas.

Reviewed by Dave Stone and Sandy Wagner

Dave Stone teaches at Rollingwood Elementary School in San Bruno, CA 94066, and has provided microcomputer support to classrooms in grades one through five since 1976. He is an active member of CUE (Computer Using Educators).

Dr. Sandy Wagner teaches math and computer programming at Mountain View High School, Mountain View, CA 94041. He is president of Computer Using Educators, a group dedicated to keeping computer teachers in touch with each other and with new developments in the field.



SWORDQUEST
 Fantasy Games Software
 P.O. Box 1683
 Madison, WI 53701
 \$12.95, Cassette Tape (PET)

In our library project that is part of ComputerTown, U.S.A!, we have a PET computer (several additional machines are on their way) and a file of cassette tapes. The program most often checked out by the kids that frequent the library project is Swordquest.

During a brief two week period, when we had to order a second version of the fantasy simulation, the kids nearly drove everyone crazy asking for the game. (The original cassette probably got damaged by handling and sticky fingers touching the surface of the tape. The tape itself is protected; you cannot make copies with the SAVE command.)

Now we have several new copies. The Swordquest fever is once again at its peak. Why do the kids play this game so much?

Because it's fun, exciting, difficult-to-win, and challenging. The fantasy simulation contains hidden monsters, giant spiders, a sacred tomb with jewels and a magic sword, magic arrows, a bow, a regular sword, and commands that help you in your adventure through the maze of tunnels. You can *listen* for movement of the hidden creatures. You can *wait* to see how the monsters are going to position themselves, once they appear. You can *brace* yourself before entering a conflict. Bracing increases your probability of winning a battle.

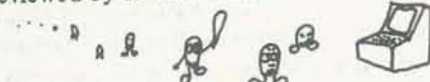
All of this action is graphically displayed on the screen. The full maze is always in view, and you know where the tomb is located. The problem is to get there safely, past the many hidden beast/guardians of the tomb. Once there, you are still not finished with the game. The monsters, sensing that you have entered the sacred tomb, begin a race to surround you to prevent your escape. Your magic sword, a treasure from the tomb, gives you some leverage as you battle your way out.

The game is well-documented, and there is an interesting background story related to the action on the screen. The action is fast-paced; the game easy to play in terms of getting started with it.

Up to this point, I have only said that kids like to play Swordquest. I don't want to leave a false impression. *Adults* are also captured by the simulation. They spend as much time with the game as the kids, if the kids will let them near the computer.

Swordquest is currently available only for the PET. It would be great to have this game on some of the other computers. But, even if you don't own a PET, you still may want to order this game, find a friend with a PET, and both of you spend a lot of enjoyable time in the world of Swordquest.

Reviewed by Ramon Zamora



WHAT IS A COMPUTER?

By Marion Ball
 Houghton Mifflin Co., 1972
 (1978 impression)

Computers and children are natural companions in the age of electronic marvels. It is adults who complain that computers are too technical, not children.

Then, where are the books for young people? Where are the books that will zoom kids into the computer age, that will help make computer addicts out of them? One such book is *What Is a Computer?* by Marion J. Ball.

This book is not only an excellent introduction to computers for children, it is also good for the aspiring adult hobbyist. It is simple without being simplistic, informative without being wordy, and surprisingly comprehensive.

Marion J. Ball is assistant director of Medical Computer Activity at Temple University Health Sciences Center. Mrs. Ball has previously taught in the Lexington, Kentucky, public school system and at the University of Kentucky. She has also written numerous articles and lectured on computer applications in medicine.

The text is easy-to-understand, although no talking down to children is done. For example: "Digital computers solve problems by counting numbers. They can add, subtract, multiply and divide... A digital computer can also compare two numbers to find out which is greater or if they are equal."

The book also includes a glossary and an excellent index, making it easy find answers to specific questions.

If you are a computer addict and have a son or daughter interested in computers, or if you work with youngsters and would like to introduce them to computers, or if you yourself are a computer novice, *What Is a Computer?* is worth your attention.

Reviewed by Louisa Jartz
 North Canton, Ohio

Louisa Jartz is a former children's librarian and secretary of the Cleveland Digital Group, the local hobbyists' club.

MELISSA & JOHN & THE MAGIC MACHINE

by T. J. Cohen & J. H. Bray
 Byte Publications, Inc., 70 Main Street,
 Peterborough, NH 03458.
 \$2.00, 15 pages/crayons.

I think the book is enjoyable.

Wait a minute. That's what all book reviews say. This is going to be the honest-to-goodness truth. You may kill me, but this is the truth. Here goes.

I think the book is enjoyable.

It is quite unlike any other coloring book I have seen. And I've seen quite a few, since most coloring books were made in the seventies, and that's when I was a kid. (And I still am.)

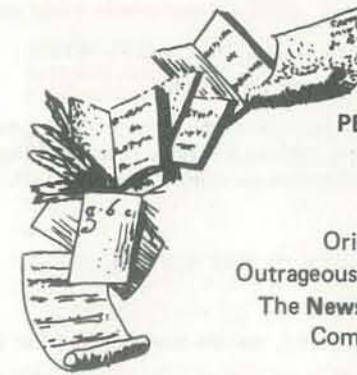
Everyone is slightly confused about "The Magic Machine," except Dad. He knows all about computers and is going to get one.

This charming (no, that's not quite right) different and nice coloring book is definitely worth the \$2.00 price. It basically is 16 pages of coloring book and six Noah's ark crayons. I think it would be a good coloring book for ages three to six. It is not very informative, but it can make a child ask questions. Who knows? Maybe your child will become a famous computer person.

As I said before, it is quite unlike any other coloring book. It is *not* your typical Snow White story. And that is final!

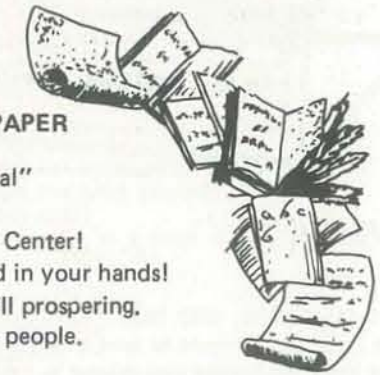
Reviewed by Rachel Wasserman, Age 9
 Palo Alto, California

collector's items



PEOPLE'S COMPUTER COMPANY NEWSPAPER

"The oldest personal computing periodical"
 Launchpad for *Dr. Dobb's Journal*
 Original publication of the walk-in Computer Center!
 Outrageous predecessor to the magazine you now hold in your hands!
 The Newspaper was a wild idea come true, a seed still prospering.
 Computers can be demystified; they can be for people.



SPECIAL OFFER

All 12 available back issues of PEOPLE'S COMPUTER COMPANY NEWSPAPER for only \$15. Over 1/3 off regular price.

People's Computer Company Newspaper

Single issue price : U.S. \$2.00, Foreign \$2.50
 Vol. 1 ('73) # 3
 Vol. 3 ('74-5) # 1,4
 Vol. 4 ('75-6) # 3,4,5,6
 Vol. 5 ('76-7) # 1,2,3,4,5

Quantities are limited. This offer good until 12/30/79 only.

Check here for special offer, or circle your choices.

Please print plainly:

NAME _____
 ADDRESS _____
 CITY/STATE _____
 ZIP _____ COUNTRY _____

My payment of \$ _____ is enclosed

NOTE: All back issue orders must be prepaid. Payments must be in U.S.\$ drawn on a U.S. bank.

OTHER MAGAZINES AVAILABLE

AT THE BACK ISSUE PRICES BELOW (Circle your choices.)

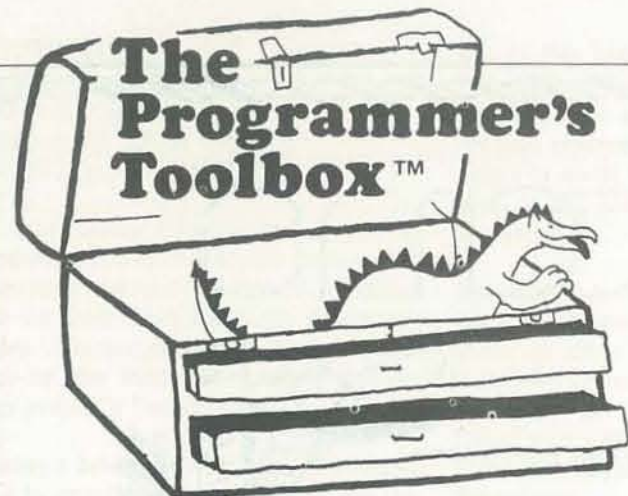
People's Computers (formerly PCC Newspaper)	U.S. & Territories	Foreign by Surface mail
Vol. 5 ('77) #6	\$2.50 ea.	\$3.00 ea.
Vol. 6 ('77-8) #1,2,3,4,5	"	"
Vol. 7 ('78) #1,2	"	"
Vol. 7 ('78) #3	\$3.00 ea.	\$3.50 ea.
Recreational Computing		
Vol. 7 ('79) #4,5,6	\$3.00 ea.	\$3.50 ea.
Vol. 8 ('79) #1	"	"

HIGHLIGHTS OF PCC NEWSPAPERS

Vol. 1 #3 ZINGO/"Going to Bid"/How to Buy an Edusystem/
 BASIC MUSIC/CHOMP
 Vol. 3 #1 A Fantasy of Future Forms/ Low-Cost Home School
 Microprocessor systems/ INCHWORM/MAZE/ Analysis of Teaser/
 Computer Illiteracy Problem by Peter Lykos
 Vol. 3 #4 Build your own BASIC/Mysterious RND/More INCH-
 WORM/ Computers & Art/BIOSIN Son of Hardware
 Vol. 4 #3 NUMBER/Tiny BASIC/Analog to Digital Conversion/
 Minuet in G Major
 Vol. 4 #4 Space Games/Tiny BASIC/Soloworks/DIDDLE/ STTR/
 More Music
 Vol. 4 #5 Games for the Pocket Calculator/SQUARES/Make
 Believe Computers/STRTRK/Huntington Computer Project
 Vol. 4 #6 Biofeedback/POUNCE/"We're Building Our Own
 Computer"/Programmer's Toolbox/ CCC/TV as Terminal and
 Game Center
 Vol. 5 #1 Low Cost Software/Tiny BASIC, Tiny Trek/ Your
 Brain is a Hologram
 Vol. 5 #2 Dungeons & Dragons/HATS/One on One/PLANETS/
 The Positive of Power Thinking
 Vol. 5 #3 STORY/SNAKE/More Build Your Own Computers/
 Introducing PILOT/FROGS
 Vol. 5 #4 REVERSE/Robots/Tiny PILOT/Space & Computers/
 Conversational Programming
 Vol. 5 #5 Z-80 PILOT/6502 Assembly Programming/Tiny BASIC
 for Beginners

Please send in this form or facsimile to: People's Computer Co.,
 1263 El Camino Real, Box E, Menlo Park CA 94025.

Only the issues listed here are available. Price includes issue, handling, and shipment by second class or foreign surface mail. Within the U.S., please allow 6-9 weeks to process your order second class. Outside the U.S., surface mail can take 2-4 months. For faster service (U.S. only) add \$.50 per issue requested and we'll ship UPS. Outside the U.S., add \$1.50 per issue requested for airmail shipment.



BY EVERYBODY

In Vol. 1, No. 3 of *PC*, 1973, Marc LeBrun began a column that provided routines that could be used as part of a "toolbox" of computer skills. We revived that column in the May-June 1979 issue of *RC*.

Here are some more "tools" for the Toolbox. Hope you can use these new programs. If you have ideas for useful routines, write them down and send them to us. —RZ

PT 7 : REWARDING A SUCCESSFUL GUESS

The simple open (in line) subroutine in lines 200-290 rewards a player who has just won a guessing game on the TRS-80. It fills the screen with about 100 stars (*), then tells how many guesses it took, then asks if the player wants to play again.

We show the reward used with a modified version of PT4 : PRINT @ SCROLLING (*RC*, Jul-Aug 1979). We have added a guess counter (NPT) to that program.

```
100 REM *** PROGRAMMER'S TOOLBOX (TM) NUMBER 7
110 REM *** RECREATIONAL COMPUTING, SEP/OCT 1979

120 CLS
130 X = RND(100)
135 NPT = 0

140 PRINT@832, "I'M THINKING OF A NUMBER FROM 1 TO 100!"

150 PRINT@960, "GUESS MY NUMBER!"; : INPUT G
155 NPT = NPT + 1
160 IF G<X THEN PRINT @ 896 + 32, "TRY BIGGER!"; : GOTO 150
170 IF G>X THEN PRINT @ 896 + 32, "TRY SMALLER!"; : GOTO 150

200 REM *** PROGRAMMER'S TOOLBOX (TM) NUMBER 7
210 REM *** RECREATIONAL COMPUTING, SEP-OCT 1979
220 CLS
230 FOR APT = 1 TO 100
240 PRINT@RND(1023), " * ";
250 NEXT APT
260 PRINT@409, "THAT'S IT!!!"
270 PRINT@531, "YOU GOT IT IN" NPT "GUESSES."
280 PRINT@657, "TO PLAY AGAIN, PRESS 'ENTER'"
290 IF INKEY$ = "" THEN 290 ELSE 120
```

The messages printed by lines 260, 270, and 280 are approximately centered on the screen and double spaced to make them readable. The stars and the messages stay on the screen until someone presses the ENTER key, or most any other key. Then, the game starts again.

You can modify this open subroutine to a closed subroutine, called by a GOSUB. Simply change line 290 to

```
290 IF INKEY$ = "" THEN 290 ELSE RETURN
```

If you do this, be sure to provide the value of NPT in your main program. And, if you relocate the subroutine by changing the line numbers, remember that our line 290 refers to itself.

```
290 IF INKEY$ = "" THEN 290 ELSE RETURN
```

If you relocate the routine, use the same line number in both places.

Although we show the subroutine in use with a number guessing game, it is suitable for most any guessing game.

BY THE DRAGON

PT 8 : ANOTHER STRING SQUEEZE

This subroutine (lines 900-1040) removes all characters from APT\$, except those characters which are in BPT\$, defined in line 920.

```
100 REM *** TEST PROGRAMMER'S TOOLBOX # 8
110 CLS
120 INPUT APT$
130 GOSUB 920 ← Call the subroutine.
140 PRINT APT$
150 PRINT
160 GOTO 120

900 REM *** PROGRAMMER'S TOOLBOX # 8
910 REM *** RECREATIONAL COMPUTING, SEP-OCT 1979
920 BPT$ = "0123456789D"
930 BPT = LEN(BPT$)
940 APT = LEN(APT$)
950 IF APT = 0 OR BPT = 0 THEN RETURN
960 ZPT$ = ""
970 FOR JPT = 1 TO APT
980 UPT$ = MID$(APT$, JPT, 1)
990 FOR KPT = 1 TO BPT
1000 VPT$ = MID$(BPT$, KPT, 1)
1010 IF UPT$ = VPT$ THEN UPT$ = ZPT$ + UPT$ : GOTO 1030
1020 NEXT KPT
1030 NEXT JPT
1040 APT$ = ZPT$ : ZPT$ = "" : RETURN

9999 END
```

It works on the TRS-80 and will probably work on most Microsoft™ BASICs. Look at line 920. In this example,

```
BPT$ = "0123456789D"
```

You, of course, can change BPT\$ to anything you want. To show you how it works, here is a RUN.

```
?0123456789D
0123456789D
```

```
?3 D 6
3D6 ←
```

```
?ABCDEFGHJKLM
D
```

```
?23%XD#7
23D7
```

Aha! You people who play fantasy adventure games know about 3D6!

Yes, this subroutine is s-l-o-w. Do you see how it works? If not, write us a letter. If we get several letters, we will publish an explanation of this or any Programmer's Toolbox tool. If we get only one or two letters, we will only respond to them with a personal Dragon answer.

BY THE DRAGON

PT 9 : GETTING AROUND "EXTRA IGNORED"

Here is a routine that works on both the Apple and the PET. It allows you to input a string of data that contains commas and colons. On most machines, when a comma or colon is encountered during an INPUT, the rest of the input line is thrown away, and the message "EXTRA IGNORED" is printed. This routine solves that problem. For the Apple, change line 40 to:

```
40 GET GS
```

```
10 REM*** PROGRAMMER'S TOOLBOX #9
20 REM*** INPUT STRINGS WITH COMMAS AND COLONS
30 PRINT "INPUT STRING: ";
40 GET GS: IF GS="" THEN 40
50 PRINT GS: IF GS=CHR$(13) THEN 70
60 AS=AS + GS : GØ TØ 40
70 PRINT A
80 END
```

Note: Line 50 is used here to verify input. If you make a subroutine out of this, delete line 50. Also, you may not want line 70.

BY JOHN W. DAVISON

PT 10 : CARD SHUFFLE

Considering some of the strange card shuffling routines I've seen, this routine deserves a wider distribution. I first saw it in *The Little Book of BASIC Style* by John M. Nevison. It is correct in the sense that it generates an arbitrary permutation out of the set of all permutations.

Assumptions:

- A(1:52) is the array containing the cards.
- RND(x,y) generates a random integer between x & y inclusive.
- All the cards are to be shuffled.

```
10 REM *** PROGRAMMER'S TOOLBOX # 10
20 REM *** CARD SHUFFLE ROUTINE
30 DIM A(52)
40 FOR I=52 TO 2 STEP -1
50 J=RND(1,I)
60 T=A(I)
70 A(I)=A(J)
80 A(J)=T
90 NEXT I
```

Note: This routine is not written for any particular machine. Line 50 will need to be changed to whatever RND function call you have on your computer.

BY ERYK VERSHEN

Reader Service

MOOVING?

IMPORTANT

Please advise us of address changes 60 days in advance and attach your magazine mailing label here. Please clip this notice and mail, or send us a reasonable facsimile.



Name (please print new information)

Address

City

State

Zip Code

Country

Recreational
COMPUTING

P.O. Box E, 1263 El Camino, Menlo Park, CA 94025

Apple II's Three M's

(Memory, Monitor, & Machine Language)

Part I

BY CHUCK CARPENTER

Chuck notes in his letter with this article that he lives in Carrollton, TX, but is a member of the Apple Corps of Dallas. Chuck, a frequent contributor to the magazine, is highly knowledgeable on the Apple. Now you get the benefit of his knowledge as he leads you through an introductory two-part series on how to use the machine language capabilities of your Apple.

The only questions left unanswered are:
Isn't Dallas a TV show?
and
Where in the world is Carrollton?

- RZ

Built into your Apple II is a powerful assembly language programming capability. To get a good foundation for programming in assembly language, an understanding of the Apple II's fundamental operation is helpful. Every function, because of internal design structure, is related to *memory*. The job of converting the things you do, to things the computer can understand, is handled by the *System Monitor*. Apple II uses a 6502 Microprocessor as the central control unit. Communications with the rest of the computer is done with a *machine language* unique to the 6502. Let's look at *memory*, *monitor* and *machine language* separately.

Memory

A total of 65536 memory cells are possible in the Apple II. If you plugged 16k memory chips into all the available expansion sockets, that's how much you would have. Not all of these memory cells are available for you to use. Some are already in use by built-in computer programs.

Two Kinds of Memory

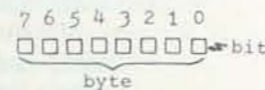
Your computer uses two kinds of internal memory. One is called RAM, the other is called ROM. (External memory is called tape or disk, but that's another story.) RAM stands for *random access memory* and ROM stands for *read only memory*. Both are actually random access. That is, a memory cell can be selected and accessed directly. It is not necessary

to pass through each memory cell in sequence to get to a certain one. With RAM, you can read from a memory cell or write to it. (RAM is also known as read/write memory.) You can only read from a ROM memory cell.

When power is turned off, anything previously stored in RAM is lost. When power is turned on, you have to restore programs in RAM. Tape players or a disk are the usual input sources for the Apple II. Changeable memory, like the Apple II RAM, is called volatile memory. ROM on the other hand is non-volatile. It is always there when power is turned on. There is another characteristic of Apple II RAM. The type used is known as *dynamic* memory. Many other computing systems use static memory chips. These require a lot more operating power. The dynamic chips use far less power but require a technique called refreshing. Once a static memory cell is activated it stays in a particular condition until specifically changed or power is lost. Once a dynamic cell is activated it has to be refreshed periodically. Otherwise, the information in the memory cell fades away. (A similar condition occurs on your video screen. If the dots on the video screen weren't refreshed, the picture would fade out. It happens so fast though, that you can't see it happening.)

Cells and Words (Bytes and Bits)

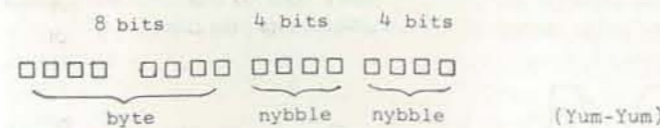
Memory cells in the Apple II can be called words because each cell can store some kind of data or instruction. It is also correct to call each memory cell or word a *byte*. Each byte is made up of 8 *bits*.



A bit gets its name from the words *B*inary *digi*T. Binary means consisting of two parts. In a computer, a bit has two states, *on* and *off*. Conventions of definition provide a technique for describing the on and off states. A '1' is used for the on state, and a '0' is used for the off state. The state of each bit is used to determine the value of a binary word or byte. From now on we will use byte to mean an 8 bit binary word.

How Many Bytes?

To make a number large enough for addressing all 65536 memory locations takes 16 bits. Two bytes are needed to do this in the Apple II. Internal to the 6502 microprocessor are several RAM locations called *registers*. One of these is a 16 bit (two byte) register used to address all the memory locations. Also, it is conventional to divide each byte into two 4 bit *nybbles*.

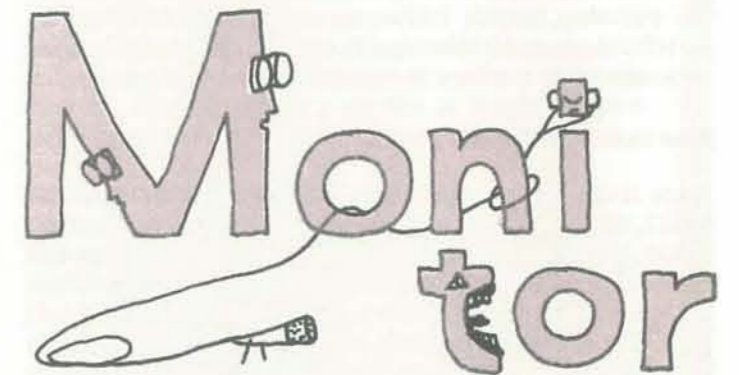


Because we are using two states to represent each bit, a value based on a power of two can be assigned to each bit. Notice that the value of the binary number is 0 when all bits are in the off state. This is important! The first number in the number scale is 0. The first address in Apple II memory is located at 0000 (4 nybbles having zero value). To find the value of a binary word, add the powers of two (called weighting) for each bit that is a '1' (on).

Binary	1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1
Weight	2^{15} ----- 2^0
Decimal =	Sum of each value for each 1 bit.
	$= 2^{15} + 2^{14} + 2^{13} + 2^{12} + 2^{11} + 2^9 + 2^2 + 2^0$
	$= 32768 + 16384 + 8192 + 4096 + 2048 + 512 + 4 + 1$
	$= 64005$

If all the bits were 'on' the calculation would have been $2^{16} - 1 = 65535$. This is the address of the highest byte in memory. Notice that the values increase from right (least significant) to left (most significant). This notation is also conventional and is used to determine the decimal equivalent of bytes and nybbles too.

There are many books that explain computer mathematics. We have covered enough for this article. A little discussion of binary and decimal numbers will be found in the next section.



Apple's System Monitor is located permanently in ROM. Your computer would not be able to do any of its computing functions without the monitor. The monitor is a collection of programs (written in 6502 machine language) provided by the computer designers. If you consider everything your computer does, you can picture the tasks performed by the monitor. Before we continue discussing the System Monitor, an understanding of another number system is needed.

Hexadecimal

Memory address identification and 6502 machine language uses a numbering system called hexadecimal. Binary numbers are part of a base 2 system. Our daily activity uses the base 10 decimal system. Hexadecimal numbers are based on 16 digits. Thus it is a base 16 number system and is represented in Table 1.

Decimal	HEX	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Table 1

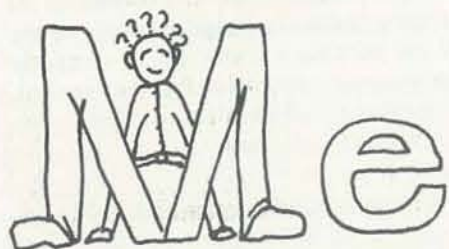
Letters are used to represent numbers 10 to 15. At first, this may be a bit confusing, but only one character is needed in hexadecimal to represent numbers up to 15. For instance, if you want to determine the HEX values for the address used in the earlier example, it would look like this:

Binary -	1111	1010	0000	0101
HEX -	F	A	0	5
Decimal -	64005			

So, the decimal address 64005 is HEX address \$FA05. (The \$ is used in front of the number to indicate that it is HEX.) Use the values in table 1 to convert each nybble in binary to its HEX equivalent. Note that it takes 4 digits in HEX, 5 in decimal, and 16 in binary to express the same number.

What Does K Mean?

Apple II monitor programs begin at address \$F800 or decimal 63487. If you subtract this value from the highest address 65535, you will get 2048. This is the number of bytes used by the Monitor ROM. In HEX, 2048 is equal to \$800 and is called 2K bytes of memory. (The term 'K' refers to a quantity of 1024 memory words or bytes.) This use of 'K' comes from scientific notation where a quantity of 1000 is known as a 'Kilo-'. Since memory is numbered in binary, the nearest quantity is 1024. Thus a 65,535 byte memory is called a 64K (64 x 1024 = 65,535) memory.



What's in It for Me?

This article started by saying that everything the Apple does is related to some part of memory. This is called memory mapping. (Refer to pages 33 and 136 in the Apple II Reference Manual. The memory maps on these pages show how the

various memory ranges are used within the computer.) By now you know the monitor is located between addresses \$F800 and \$FEE, and that the monitor programs are in ROM. Let's see what some of the monitor programs do.

- When you turn power on and press RESET you get a "beep" and a prompting character on the screen. Monitor programs do this.
- Each time you press any combination of keys, the sequence is analyzed by the monitor.
- If the keystrokes have any meaning (are valid commands) the monitor responds.
- The monitor responds if they are not valid too. You get a "beep" and another prompt telling you to do it over.
- Monitor programs *read* a cassette tape program and load the contents into memory.
- Saving programs by *writing* them onto tape is handled by the monitor too.
- Control of video output on the screen is a monitor task.
- The monitor also generates the 16 colors used with low resolution graphics.

There are many other computing tasks carried out by the system monitor. But this should be enough to point out the importance of this 2K section of memory.

Other memory uses include the Integer BASIC interpreter and other utility programs. These are permanently stored in ROM too. Addresses for BASIC and the utilities start at \$E000 and end at \$F700. The memory space from \$C000 to \$CFFF is reserved for the 8 input/output (I/O) expansion connectors. Each connector is assigned a range of memory. When you plug an expansion card, such as a printer board or clock, into one of these connectors, it operates from the assigned range of memory for the slot it is in. (Applications are described in detail in the Apple II Prototyping Manual and in the Reference Manual.) Memory addresses for controlling the speaker and the game paddle connectors are in this area too. The space from memory location \$D000 to \$DF77 is reserved for ROM expansion.

Work space for various computing functions is assigned to RAM in low memory. Also, the area of memory that represents all the possible character positions on the screen are in RAM. This uses up most of the bytes of memory from \$000 to \$07FF. Memory used by all the computing functions named so far has taken more than 18K bytes. The RAM space left for you, the user, is from \$0800 to \$BFFF. This represents over 47000 bytes (if you have 16K memory chips plugged into all the memory sockets) of memory. Your programs can use some of this memory, and we will discuss how in the next section.

Hope you liked this introductory section. Chuck will be back next issue with the third M, machine language. He will lead you through the magic mystery tour of machine mnemonics and other marvelous machinations. See you then.

More to come

NEW HAMPSHIRE'S INTREPID HOBBYIST

Dear **COMPUTER FREAKS:**

I read *Dr. Dobb's* and *Recreational Computing*, and other than the Heath Users' Group, you are my only contact with the real computer world. I have saved my pennies and own a Heath H8. It has 16K of memory and has both serial and parallel interface cards but as yet no terminal. I have spent time and money trying to interface various cards to my system but have had little luck.

Two years ago, I had never played Star Trek or even sat at a terminal. My contact with computers seemed limited to bank statements and visits to science museums. I was, however, an electronics technician and realized that unless I got involved soon, I would be unemployable as the new technology came in. Since I am legally blind, I have a hard time getting work anyway.

When I finally got into computers, I was living in Massachusetts. A friend bought an SWTPC 6800 and had it running with a terminal in six months. He promptly got into the game thing and was soon hooked on Star Trek. When I visited him, I worked on new games or in changing the ones he had. Another friend bought a Radio Shack TRS-80. He didn't need it but did it to keep up with us. I wrote my share of programs on it before I moved.

Recently, I have had only my H8 with no terminal or printer or anything. I have run several programs using only the front panel and have learned enough to use the machine as is. I am currently working on a program for

inventory of a small hobby store, using just the keypad and readouts, where an item number can be entered and quantities read out, altered, and prices adjusted. Another function will allow reading out the entire contents for end-of-the-month inventory and ordering. It will have the ordering status indicated also, and all data will be stored on tape.

If it were not for the games I played on other systems, I would not be able to learn as fast as I have. For some of us, games do have an important purpose, just as reading science fiction does. They make things we bear a little easier. If zapping Klingons is a release and turns you on, fine. If piloting a starship in imaginary space replaces my lack of ability in driving a car, great.

As I read over what I have written, I again come back to the fact that I don't have a terminal for the computer. Perhaps someone has a spare terminal they could donate, or perhaps some company wants to field test a unit? If I could get a terminal, I might come up with some very challenging games, since I have plenty of time to spend on programming. Perhaps I could get a disk system next year.

As a last note, please keep up the good work and continue to give us the best magazines. I like the changes and enjoy every issue.

Yours truly,
Robert Howarth, Jr.

RFD #1 Box 36
Lisbon, N.H. 03585

Announcements

Hardware

Dual Drive for TRS-80. Percom Data Company is now selling a dual disk drive for the TRS-80. Called the TFD-1000, the unit provides 800K bytes of on-line storage; two systems (four drives) may be used to provide 1.6M bytes on line. The MICRODOS operating system, which replaces TRSDOS, provides full random access capability and requires less than 7K of RAM. MICRODOS is supplied on a system diskette that includes BASIC program examples. The TFD-1000, complete with cable, operating system, PAM card and documentation, costs \$2,495. For further information, contact Percom Data Company, 211 N. Kirby, Garland, Texas 75042. (214) 272-3421.

Light Pens. The 3G Company of Gaston, Oregon, offers light pens for both the TRS-80 and the PET 2001. Since the light pens allow direct interaction with information on the CRT screen, they make programs accessible to persons with no computer background. They can also be used to create a number of unique games and special graphics effects.

The light pen for the PET 2001 comes completely assembled and with a sample program. It sells for \$29.95 (plus \$1.50 for postage and handling within the U.S.; \$6 for foreign orders). The light pen for the TRS-80 costs \$34.95. For more information on either product, contact 3G Company, Route 3, Box 28A, Gaston, Oregon 97119. (503) 662-4492.

Full-sized Floppy. An eight-inch floppy disk drive is now available for the TRS-80. The new product from Parasitic Engineering is called Maxi-Disk. Used with an option called the Shuffleboard, it can run CP/M and TRS-DOS at the same time. (This piggy-back board plugs into the Z-80 socket and remaps memory under software control, allowing CP/M to run in its standard location while

the lower 16K of memory are shuffled to high memory and all other locations are shuffled down) Parasitic points out that this development makes the large library of CP/M programs available to the TRS-80 user.

The full-sized floppy disk with controller board sells for \$995. The Shuffleboard option, provided with CP/M on an eight-inch disk, sells for \$245. For further information, contact Parasitic Engineering, P.O. Box 6314, Albany, CA 94706. (415) 527-6133.

Card Reader. Chatsworth Data Corporation has released a mark sense card reader designed for test scoring on personal computers. Called the MR-500, the unit comes with a test scoring program which does item analysis, plots histograms of test score distribution, provides the mean and standard deviation, and gives raw scores. Interfaces are presently available for the TRS-80, Apple II, and PET, with others under development. For more information, contact Chatsworth Data Corporation, 20710 Lassen St., Chatsworth, CA 91311. (213) 341-9200.

Micro Printer. The new Trendcom 100 Intelligent Printer features 40-character-per-second printing with a 96-character set. It offers full line buffering and bidirectional look-ahead printing; after one line has been printed, left to right, the internal microprocessor

examines the next line, then moves the print head to the last character of the line to be printed and prints right to left. The 5x7 dot matrix provides clear copy on white paper. Interfaces are available for the TRS-80, Apple II, PET, and Sorcerer. The Trendcom 100 is available in most retail computer stores for about \$375. For more information, contact Trendcom, 484 Oakmead Parkway, Sunnyvale, CA 94086. (408) 737-0747.

Stringy Floppy. A "stringy floppy" storage system for TRS-80 and SWTPC computers has been introduced by Exatron Corporation. The TRS-80 version of the system consists of a small freestanding module enclosing the drive unit, control electronics, and firmware; a sealed-unit power supply for the AC outlet; and a ribbon connector to the TRS-80. The SWTPC model consists of a freestanding drive module, a controller board mounted in the computer motherboard, and a connecting cable. The individual continuous-loop tape wafer, less than a fourth the bulk of the standard audio cassette and holding up to 40K bytes, is inserted in the slot in the front of the drive module. All operations are software controlled; the utility programs are stored in the firmware. The TRS-80 unit costs \$199.50. For more information, contact Exatron, 355 Ryder Street, Santa Clara, CA 95051 (408) 737-7111.

Software

8086 BASIC. Microsoft has just released a new version of BASIC for the 8086 16-bit microprocessor. According to Microsoft, BASIC-86 is completely language-compatible with the current release 5.0 of standard Microsoft 8080 BASIC. This means users of 8080 BASIC can upgrade to an 8086 microprocessor without modifying existing programs. BASIC-86 is available in two versions: Extended and Stand-alone Disk, both for Intel SBC 86/12. Prices are \$350 for extended; \$600 for disk. For more information, contact Microsoft, 10800 NE 8th, Suite 819, Bellevue, WA 98004. (206) 455-8080.

Apple-80. 8080 programs can now be run on 16K or larger Apple IIs, thanks to this 8080 simulator. Apple-80 can also be used as a design and debugging aid in developing 8080 software. According to its creator, Apple-80 provides single-step, trace, and run modes and executes all valid 8080 op-codes. 6502 subroutines can be called directly from 8080 programs, allowing full access to Apple monitor and user-written functions; conversely, 8080 routines can be embedded in 6502 programs.

The Apple-80 package includes Apple-80, a manual, an 8080 program which demonstrates the features, and an Apple-80 reference card. Price is \$20 plus \$1.50 for shipping and handling. (California residents must add 6% tax.) Order from Dan McCreary, Box 16435-D, San Diego, CA 92116. (714) 281-5758.

Poison Control. A potentially life-saving program, offering emergency advice in the event of accidental poisoning in the home, is available on North Star Diskette. The package by Roger O. Littge, M.D., is written in BASIC and uses word recognition

to identify household substances; maximum search time is six seconds. Home Poison Control is available on North Star Diskette with manual and complete source listings for \$28 or on a CBASIC version eight-inch diskette with manual for \$32. Contact Berkeley Medical Data Associates, P. O. Box 5279, Berkeley, CA 94705.

Abridged Adventure. Scaled-down versions of Adventure for the H8 Heath computer have been announced by Eggert Engineering. They are available on cassette in 24K and on disk in 16K. "We took our 32K version and deleted two of the treasures, one of the mazes, eliminated the hints, and used only the long description of each location. The remaining game logic and text was unchanged," reports designer John Eggert. Both the 32K and 24K versions are available on cassette (with instructions) for \$16.50 each. Non-Heath users can get both versions on 8-level binary paper tape, with modification instructions, for \$65 each. A disk version has also been developed that will run under HDOS on an H8 with 16K of RAM. Price on the disk is also \$16.50. Order from Eggert Engineering, 95 Adams Drive, Stow, MA 01775. (Massachusetts residents must add 5% tax.)

Video Loom. Weavers can now try out designs quickly and easily on a computer via a new Apple II program called Video Loom. The program simulates a loom with up to 70 harnesses and allows the weaver to choose the colors, yarn sizes, treadling order, etc. Once the loom is "warped" and the weaving sequence established, the computer weaves a full-color picture of the design; changes can be made in the pattern before it is completed. It can then be saved on diskette. Video Loom is available on a five-inch diskette for 32K Apple IIs. The price is \$49.95 plus \$2 for postage and handling. Californians should add 6% sales tax (\$3.25). Contact Howard Harowitz, Systems for the Arts, 1510 Grant Street, Berkeley, CA 94703.

Satellite Tracking. You can turn your micro into a satellite tracking site or space center with three new programs available from Sat Trak International. Written in BASIC, these programs can be used on Apple IIs, Sorcerers, or TRS-80s with at least 16K of RAM. The programs enable the user to track satellites anywhere, locate them at night with the naked eye, and predict future sightings. For more information, contact Sat Trak International, c/o Computerland of Colorado Springs, 4543 Templeton Gap Road, Colorado Springs, CO 80909. (303) 574-4150.

Apple Library. With the Apple II Software Library, Volume I, you get an assortment of 16 programs ranging from games to teaching exercises to music and graphics demos. The programs are all written in Integer BASIC and come on a cassette that sells for \$14.95. Volume II of the Apple Software Library has similar variety in its 17 programs, ranging from a math test to computer-generated poetry to music by Bach. This cassette is also \$14.95. Order the tapes from the Soft-One, 315 Dominion Drive, Newport News, VA 23602.

Aircraft Simulator. For earth-bound pilots, this may offer the next best thing: a program with engine sound effects, high resolution color graphics, and an instrument panel that shows—on the TV screen of an Apple II—attitude, air speed, altitude, rate of climb and descent, compass heading, and rate of turn of the aircraft. The instruments respond smoothly and in real time to the keyboard and game paddles which control the "aircraft." An ideal stand-alone program, it can also be used as the basis for more sophisticated flight simulators. The program is written in both Integer BASIC and machine language and is designed to run on an Apple II with at least 16K of memory. Tape and instructions are \$8.95. Order from the Soft-One, 315 Dominion Drive, Newport News, VA 23602.

Users' Groups

Northwest PETs. The Northwest Pet Users Group, based in Portland, is trying to locate PET buffs in Oregon and Washington. If you'd like to meet with other PET folks, contact John F. Jones of the NW Pet Users Group, 2134 NE 45th Ave., Portland, OR 97213. (503) 281-4908.

Honolulu Apples. HAUS (Honolulu Apple Users Society) meets the first Monday of each month at the Computerland store in Honolulu. The group has a newsletter containing Apple program tips and techniques, listings, reviews, etc. There are also weekly sessions for learning programming and exchanging information. HAUS is interested in exchanging news with other Apple groups. Contact Bill Mark, 98-1451-A Kaahumanu Street, Aiea, Hawaii 96701. (808) 488-2026.

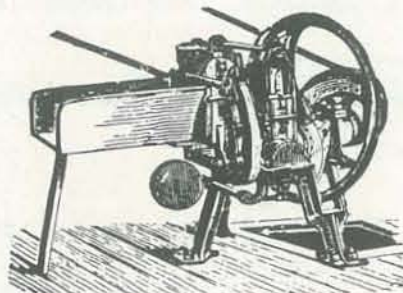
British Hobbyists. The North London Hobby Computer Club (NLHCC) now has more than 300 members, making it one of the largest such groups in Europe. There are regular monthly meetings, one-session courses, and four ongoing workshops. The New Games workshop has produced significant software, according to the NLHCC report. The club meets at the Department of Electronic and Communications Engineering in the Polytechnic of North London, Holloway, London N7 8DB, Telephone 01-607 2789. For further information, contact the club secretary at the above address.

Publications

Calculator Book. A new 12-page illustrated booklet comparing the logic systems of advanced handheld calculators is available free from Hewlett-Packard. It should be particularly helpful to educators teaching calculator usage as well as to calculator shoppers evaluating equipment; it can also help experienced users better understand their logic systems. *Advanced Calculator Logic: A Comparative Analysis* is available by writing Inquiries Manager, Hewlett-Packard Company, 1507 Page Mill Road, Palo Alto, CA 94304.

Applesoft II Manual. A complete programming reference manual for the Applesoft II language is now available from Apple Computer, Inc. The manual fully describes the extended programming capabilities offered by Applesoft II Floating-Point BASIC. Price of the manual is \$6.95. For more information, contact Jean Richardson at Apple Computer, 10260 Bantley Drive, Cupertino, CA 95014.

A Different TRS-80 Journal. From Washington, D.C., comes a different TRS-80 publication—"one for the owner or user interested in more than BASIC." Called *Insiders: the TRS-80 Hardware Journal with Machine Software*, the publication offers articles on machine language programming, hardware modifications, and other computer languages. Recent articles have included a discussion of the differences in Level II ROMs, how to get sound effects and music without hardware modification, new languages for the TRS-80, reviews of printers and disk drives, etc. Subscriptions to *Insiders* are available at \$7.50 for six issues through Computer Cablevision, 2617 42nd St. NW, Suite 2N, Washington, D.C. 20007. (202) 337-4691.



Didactic Programming. This journal of calculator-demonstrated math instruction contains articles on theory analysis, programs, reviews, and teaching methods. Subscription rate for the fall-winter-spring periodical is \$5. A sample issue is available for free. Write Educational Calculator Devices, Box 974, Laguna Beach, CA 92562. (714) 497-3600.

TRS-80 Newsletter. If you are interested in an article on data base manager, send for the July issue of the TRS-80 club newsletter from Arlington, Massachusetts, and enclose \$1 with a self-addressed stamped envelope. Write to TRS-80 Newsletter, 96 Dothan Street, Arlington, MA 02174.

BASIC Master Index. Falcon Publishing Company offers a master index to computer programs in BASIC that have appeared in all the major computer

magazines. In addition to indexing the titles and organizing them into 72 categories, the publication contains hundreds of reviews of programs, telling you clearly what the program does—and whether or not you'd want to try it. *Belais' Master Index to Computer Programs in BASIC* is available from Falcon Publishing, P. O. Box 688, Ben Lomond, CA 95005.

SAM76, 2nd Ed. The second edition of the SAM76 language manual is now available for \$15. It incorporates material published in magazines during 1978. There is also a limited number of 64-page inserts which can be used to update the first edition of the manual; they are \$5 each. A SAM76 bilingual (French-English) version of Adventure is available on eight-inch disk; an English-only version on five-inch disk. Price for the Adventure disks is \$20 each postpaid. Send orders to SAM76, Box 257, RR1, Pennington, NJ 08534.

TRS-80 Software Source. The summer edition of the TRS-80 Software Source contains 3,000 listings of TRS-80 software available from 200 vendors. Program listings are cross-indexed and divided into five sections: listings alphabetized by supplier, by subject, by BASIC and memory, by disk, and by supplier's name. Subscriptions to the Software Source cost \$10 per year (three issues); \$4 for a single issue. Order from Box 1664, Lake Havasu City, AZ 86403.

Northeast Computer Show. The largest personal and small business computer show in New England will take place in Boston, Sept. 28-30 at Hynes Auditorium. There will be futuristic displays and exhibits, including the office of the future, the computerized kitchen, and an executive mobile office. There will also be a "rookery of robots." Radio Shack, Commodore, Heathkit and other micro computer companies will demonstrate their 1980 systems.

Virginia Fleas. The fourth annual Tidewater Hamfest Computer Show and Flea Market will be held in Norfolk, Virginia, Oct. 20 and 21. The exhibit will be in Norfolk's Cultural and Convention Center; Flea Market tailgating space is available as well. The event is sponsored by Tidewater Radio Conventions, Inc., a coalition of six ham radio clubs and one computer club, DIGIT. A highlight of the weekend will be a dinner cruise on the *Spirit of Norfolk*. For tickets and information, contact TRC, P. O. Box 7101, Portsmouth, VA 23707.

Get Published! If you've written a good, usable, original program (no copying, please), for the TRS-80, send it to the Computer Information Exchange—and you may see your work immortalized on tape! The program can be in BASIC, assembly language, machine code, FORTH, PASCAL, FORTRAN, or any other language for the TRS-80. If CIE accepts your program, you will be sent two copies of the release print containing it; since there are often 75 programs on a single cassette, you will be getting a return of 150 for one. Send your submissions to People's Software, Computer Information Exchange, Box 158, San Luis Rey, CA 92068.

Bay Area Swap. The fourth annual California Computer Swap Meet will be held on September 15, from 9 a.m. to 5 p.m. at the San Mateo County Fairgrounds, just south of San Francisco. Buyers and sellers of personal computing products will be coming from throughout the West. Admission is free (though Fairgrounds parking is \$1). Both individuals and companies are invited to call John Craig, Editor of *Creative Computing*, at (805) 735-1023 for booth prices and availability. Write to: RFD Box 100D, Lompoc, CA 93436.

Other

THIS PUBLICATION IS AVAILABLE IN MICROFORM



Please send me additional information.

University Microfilms International

300 North Zeeb Road
Dept. P.R.
Ann Arbor, MI 48106
U.S.A.

18 Bedford Row
Dept. P.R.
London, WC1R 4EJ
England

Name _____
Institution _____
Street _____
City _____
State _____ Zip _____

RESPONSES TO THE NEWETT AWL CHALLENGE

By press time we have received three reader solutions to the Newett Awl goat problem.

The early postmark goes to Robert Kinghill, 5316 Grand Lake, Bellaire, TX 77401. His program is listed here.

The two other responses came from: Robert E. Healey, Ayer Rd., Harvard, MA 01451, and John Heidema, Tougaloo College, Tougaloo, MS 39174.

Everyone mentioned how much they like this kind of problem. So, we will put Newett to work on more challenges for you. I wish there were space to also print the letters that came with the solutions. They were each outstanding. If only we had a bigger magazine . . .

-RZ

```

30 REM*** "LAWN & THE GOAT"
40 REM*** BY ROBERT A. KINGSHILL
50 OPEN#3:4:CMD3
100 INPUT"DIAMETER OF LAWN=";DL
105 PRINTDL
110 INPUT"PERCENTAGE OF LAWN GOAT IS TO EAT=";PL
115 PRINTPL
120 K=0:R1=1:R2=2:R3=1.5:C=PL*PI/100
130 FR=SOR(1-R3*R3/4):F=R3*R3*ATN(2*FR/R3)+2*ATN
140 IF F=0 GOTO 200
150 IF F>0 GOTO 180
160 IF F+K>0 GOTO 200
170 R1=R3:R3=(R1+R2)/2:GOTO 130
180 IF F-K<0 GOTO 200
190 R2=R3:R3=(R1+R2)/2:GOTO 130
200 PRINT:PRINT"MAKE THE ROPE";R3*DL/2:"UNITS
210 PRINT#3:CLOSE 3
DIAMETER OF LAWN= 20
PERCENTAGE OF LAWN GOAT IS TO EAT= 50
MAKE THE ROPE 11.5872847UNITS LONG
  
```

ADVERTISING SPACE AVAILABLE



1/4, 1/2, Full Page
Inside
Front & Back
Covers

PLEASE SEND
FOR RATE CARD

Advertising Manager
People's Computer Company
1263 El Camino Real, Box E
Menlo Park, CA 94025
(415) 323-3111



TRS-80 SOFTWARE SOURCE

3,000 TRS-80 PROGRAM LISTINGS
LISTINGS BY BASIC AND MEMORY
LISTINGS BY CASS AND DISK
ALPHABETIZED BY SUBJECTS
ALPHABETIZED BY PROGRAMS
CROSS INDEXED

200 VENDOR NAMES & ADDRESSES
PUBLISHED 3 TIMES A YEAR
SPRING, SUMMER AND FALL
SUBSCRIPTION \$12.00 A YEAR
SINGLE ISSUE \$5.00 EACH
FOREIGN ADD \$2.00 EA/AIR MAIL

SOME OF OUR CUSTOMERS INCLUDE:

IBM	PEPSI COLA	US NAVY	DOCTORS
AT & T	UNION CARBIDE	US AIR FORCE	ATTORNEYS
NBC-TV	DETROIT STEELE	PROCTER & GAMBLE	TEACHERS
IT & T	TEXAS INSTRUMENTS	NAT BANK OF KUWAIT	INDIVIDUALS

no charge to vendors to list software — dealer discounts available

CHECK OR MONEY ORDER PLEASE

COMPUTERMAT Box 1664R Lake Havasu City, AZ 86403 602-855-3357