

Advanced Programming On the Atari ST

COMPUTE!

\$3.00
July
1986
Issue 74
Vol. 8, No. 7

\$3.95 Canada
02193
ISSN 0194-357X



The Leading Magazine Of Home, Educational, And Recreational Computing

Screen Handler For Commodore 64 & 128

Enhanced Input
For BASIC Programs

Hex War

Strategic Simulation
For Commodore 64, 128,
Amiga, Atari, Apple,
And IBM PC/PCjr

Screen Machine II

Advanced Drawing
Program For IBM PC/PCjr

Sound Development System For Atari 400/800, XL, XE

User's Poll:

The Top Five
FREE Programs
For Your Computer

C And The 68000

An Overview Of
This Popular Language

Apple Catalog Sorter
For ProDOS Disks





HOW PEOPLE WITH COMMON INTERESTS FIND AN INTERESTING COMMON GROUND.

Presenting CompuServe Forums. Where people from all over get together, without even leaving home.

Now thanks to CompuServe Forums, computer owners are sharing common interests by talking to each other through their computer keyboards. *Software users, computer enthusiasts, ham operators, french cooks, fire fighters, science fiction lovers and other special interest groups* are already in touch, online.

Because when you subscribe to CompuServe, you're able to reach people who want to talk about the things you do. As many people as you like. For as long as you like. Whenever you wish.

Join a conversation already in

progress or start one on your own. Ask questions. And get answers.

All it takes is a modem, most any personal computer and CompuServe.

Forum members across the country are as close as a local phone call.

You can go online with just a local call in most major metropolitan areas. And normal usage fees for weekday nights and weekends are just 10¢ a minute

Of special interest to all Forum participants is software that's FREE for the taking.

Public domain software. For all sorts of activities, from games to business programs. And it's just as easy to copy a piece of software as it is to participate in a Forum.

Become a CompuServe subscriber and get a \$25 Usage Credit to start you off.

Becoming a subscriber is as easy as contacting your local computer dealer. Or you can call us and order direct. Suggested retail price is \$39.95.

And if you'd want more information about CompuServe, we'll be happy to send you a free brochure. Because with all that CompuServe offers—we think it's in your best interest.

CompuServe®

Information Services, P.O. Box 20212,
5000 Arlington Centre Blvd., Columbus, OH 43220

800-848-8199

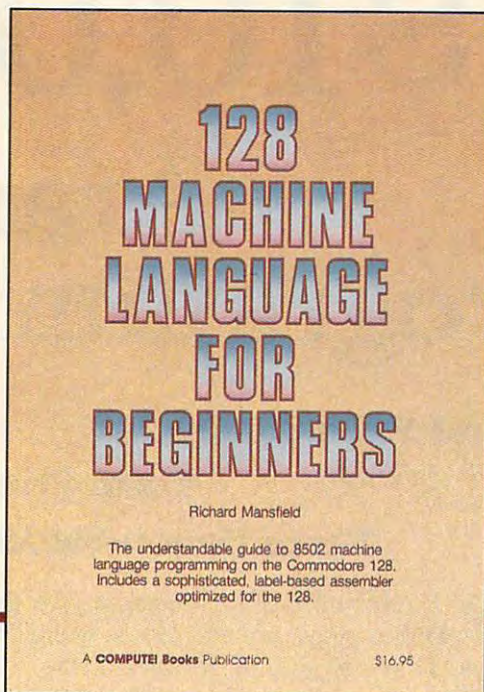
In Ohio, call 614-457-0802

An H&R Block Company

 www.commodore.ca

TAP THE POWER of the Commodore 128

By the author of
*Machine Language
for Beginners* and
*Second Book of
Machine Language*



128 Machine Language for Beginners

Richard Mansfield

One of the bestselling computer books ever has now been completely revised for the Commodore 128. Most commercial software is written in machine language because it's far faster and more versatile than BASIC. This new edition of *Machine Language for Beginners* is a step-by-step introduction to 8502 machine language programming on Commodore's 128 computer.

The book includes everything you need to learn to effectively program the 128: numerous programming examples, memory management tutorials; a complete description of the many Kernal routines and other new 128 features; numerous hints and programming techniques; and a dictionary of all major BASIC commands and their machine language equivalents. It also includes a high-speed, professional-quality, label-based assembler, optimized to take advantage of the speed and extra memory of the 128.

0-87455-033-5

\$16.95

Like the other top-quality books from COMPUTE!, *128 Machine Language for Beginners* brings you ready-to-use information in a clear, lively style that makes learning easy and enjoyable, whether you are a beginner or an advanced computer user.

An optional disk is also available which includes the assembler and example programs in the book. The *128 LADS Disk* is fully tested and ready to load on the Commodore 128. It costs only \$12.95 and saves you hours of typing time.

Order your copy of *128 Machine Language for Beginners* and the *LADS Disk* today. Call toll free 1-800-346-6767 (in NY 1-212-887-8525) or mail your payment (plus \$2.00 shipping per book or disk) to COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

COMPUTE! Publications, Inc. 

One of the ABC Publishing Companies
825 7th Avenue, 6th Floor, New York, NY 10019
Publishers of COMPUTE!, COMPUTE!'s Gazette, COMPUTE!'s Gazette Disk, COMPUTE! Books, and COMPUTE!'s Apple Applications.

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England and in Canada from Holt, Rinehart, & Winston, 55 Horner Avenue, Toronto, ON M8Z 4X6.

 www.commodore.ca

\$10,000.00

Atari ST Programming Contest!

First Prize \$5,000.00

Second Prize \$2,500.00

Third Prize \$1,000.00

Three Honorable Mentions \$500.00 each

COMPUTE! Publications, Inc. is looking for the very best original software for the Atari ST series computers. And to prove we're serious, we're offering a total of \$10,000.00 in prize money to the top six winners. That's \$5,000.00 for First Prize, \$2,500.00 for Second Prize, \$1,000.00 for Third Prize, and \$500.00 each for three Honorable Mentions. In addition, the winners will receive our standard royalties when their programs are published. And even if your program doesn't win a prize, you can still earn purchase fees and royalties if we accept your entry for publication.

Interested? If so, read these rules:

1. Entries must be your original work, previously unpublished. All those whose programs are accepted will be required to affirm this in writing.
2. You can submit as many entries as you want, but we cannot consider programs which have been entered in other contests or submitted for publication elsewhere at the same time.
3. The deadline is October 1, 1986. All entries must be received at our offices by this date. Programs submitted after this date will still be considered for publication, but will not be entered in the contest.
4. Entries are allowed (and encouraged) in virtually all software categories: home and business applications, education, recreation, telecommunications, graphics, sound and music, utilities, and desk accessories.
5. Entries may be written in any programming language—including BASIC, Logo, C, machine language, Pascal, Modula-2, Forth, FORTRAN, and Prolog—as long as they meet two requirements. First, if you're using a compiled language, the compiled object or run-time code must be a self-standing program that can be run by someone who doesn't own a copy of the language. (Exceptions are ST BASIC and Logo. Since these languages come with the ST, it can be assumed that everyone owns a copy.) Second, we must be able to legally distribute the program without incurring licensing fees or other obligations to the maker of the language. If you're not sure whether a certain language qualifies, contact its maker for clarification.
6. Entries must be submitted on a single- or double-sided 3½-inch ST disk with both the run-time code and source code included.
7. Entries must be accompanied by an article which explains how to use the program, what it does, and so on. If your program employs any new or unusual techniques that you think will be of interest to other ST programmers, you can also describe how the program works.
8. Submissions which do not win a prize and are not accepted for publication will be returned only if accompanied by a self-addressed, stamped mailer.
9. All judging will be handled by the staff of COMPUTE! Publications, Inc. All decisions regarding contest entries and acceptances will be solely at the

discretion of COMPUTE! Publications, Inc., and all decisions are final. This includes decisions regarding creativity, similarity among entries, and so forth.

10. Winners will be announced by COMPUTE! Publications, Inc. in late 1986.

11. This contest is void where prohibited by law. Full-time, part-time & previous employees of COMPUTE! Publications, Inc., and Capital Cities/American Broadcasting Corporation are ineligible for the contest, but may still submit work for publication at standard rates.

Every Contest Entry Must Contain This Form:

I warrant that the program presently entitled _____
_____ is my own original work
and that the work has not been submitted for consideration elsewhere, nor
has it been previously published. If my work is accepted by you, I under-
stand that your decision as to the selection of winners and awarding of
prizes is final and without recourse on my part. I agree, should you select
my submission, to sign your standard contract, which includes assignment
of the copyright of the program to COMPUTE!, and to allow you to use my
name and image in promotional materials and other forms. (If you are under
age eighteen, your parent or legal guardian must sign for you.)

Address entries to:
ATARI ST CONTEST
COMPUTE! Publications, Inc.
P.O. Box 5406
Greensboro, NC 27403

COMPUTE! Publications, Inc. 
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

Announcing major news for Atari ST users:

Special
BONUS
First Issue FREE
with Subscription

COMPUTE!'s

Atari ST

DISK & MAGAZINE

A bimonthly magazine devoted exclusively to Atari ST enthusiasts that includes a disk containing all of the programs found in each issue.

Atari has proven the pessimists wrong. The Atari 520ST and 1040ST have become the bestsellers among the new generation of personal computers. Both are breakthroughs in price and performance, and the community of ST owners is growing by thousands each month.

That's one reason why COMPUTE! Publications is announcing a new magazine specially designed for ST users. At the same time, we recognize that the power of the ST presents a unique challenge to magazines which publish program listings. That's why we're including a 3½-inch disk that contains every program found in each issue—ready to load and run. No more typing!

Here's what you'll get in every issue of **COMPUTE!'s Atari ST Disk & Magazine**:

- **Top-quality programs.** Utilities. Games. Educational programs for youngsters. Application programs for home and business. And since all the programs will be on disk, there are few limitations on length or languages. A typical disk might contain an elaborate adventure game written in BASIC, a programming utility written in machine language, a dazzling graphics demo in compiled Pascal, and a useful home or business application written in Forth or C.

- **Neochrome of the Month.** Take a look at what computer artists are doing with the Atari ST. Each issue's disk contains a *Neochrome* picture file ready for you to load and admire. Are you an artist yourself? Send us a picture of your own, and we'll pay you if it's published.

- **Regular columns.** If you're a programmer—or would like to be—you'll love our columns on ST programming techniques and the C language. Or check out our column on the latest events and happenings throughout the ST community. Or send your questions and helpful hints to our Reader's Feedback column.

- **Reviews.** Honest evaluations of the latest software and hardware for the Atari ST.

- **News & Products.** A comprehensive listing of the newest releases for your ST.



- **And more:** Interviews with ST newsmakers, reports on the latest industry trade shows, and overviews of significant new product introductions.

Starting with the October issue (available September 1), **COMPUTE!'s Atari ST Disk & Magazine** will be found on newsstands nationwide for only \$12.95 per copy, including disk. Or it can be delivered directly to your mailbox six times a year for only \$59.95—a savings of over 20 percent.

As a special bonus, if you order a prepaid subscription before August 1, you'll get the first issue absolutely free!

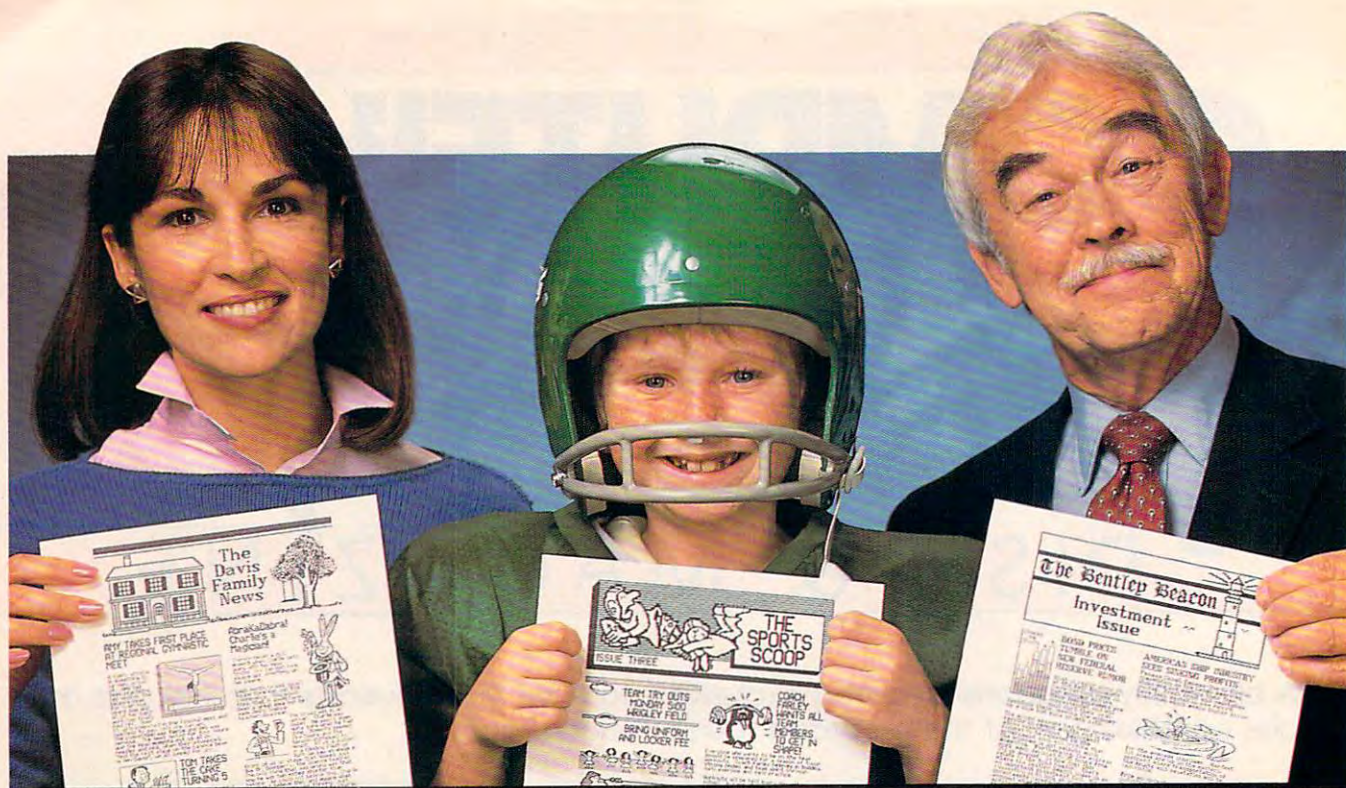
To order, call 800-346-6767. In NY 212-887-8525 or send check or money order to **COMPUTE!'s Atari ST Disk & Magazine**, ABC Consumer Magazines, Inc., Circulation Dept./8th Floor, 825 7th Avenue, New York, NY 10019.

COMPUTE! Publications, Inc. 

Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

Atari is a trademark of Atari Corporation

www.commodore.ca



The Newsroom™

Design, Create and Print Out Your Very Own Newspaper.



Look who's making news.

The Newsroom™ is an exciting program complete with everything you need to create fun and sophisticated newspapers that are uniquely yours.

Families, schools and businesses everywhere are using The Newsroom to tell their story.

Here's what you can do.

The Newsroom is surprisingly easy to use. You're in control of all design elements, starting with over 600 pieces of delightful

clip art included in the program. Combine pieces, alter them, or create your own

images with The Newsroom's dynamic drawing and editing tools. Whatever the nature of your news, you'll find art that fits the situation.

You can enter text with The Newsroom's word processor in any of five different fonts that automatically wrap around pictures. You can print out on legal or letter size paper using any popular computer printer. And, if you have a modem, you can send and receive newspapers between other owners of The Newsroom, no matter what computer they're using.

Now Available for
Apple II+, IIe, IIc
IBM PC
Commodore 64/128

More clip art to add to your collection.

Clip Art Collection Volume 1™ offers an additional 600 pieces of art in a variety of categories. You'll find the perfect piece to illustrate even the rarest of topics.

Clip Art Collection Volume 2™ offers over 800 pieces of art for businesses large and small.



*The Newsroom
is for everyone.
Look for it at
your favorite
software store.*



SPRINGBOARD

The Newsroom: Commodore 64/128—\$49.95, Apple II+, IIe, IIc & IBM PC—\$59.95, Clip Art Collection Volume 1—\$29.95 All formats, Clip Art Collection Volume 2—\$39.95 All formats.
Springboard Software, Inc. • 7808 Creekridge Circle • Minneapolis, MN 55435 • (612) 944-3915.

www.commodore.ca

JULY
JUNE 1986
VOLUME 8
NUMBER 7
ISSUE 74

GUIDE TO ARTICLES AND PROGRAMS

- 128/64/AT/AP
PC/PCir/AM

64
ST/AP
AP/AT/64
AP

•
•
•
•
•
•
•
•
AT
ST
TI
AM
PC/PCir

64
AT
PC/PCjr
ST
AT
PC/PCjr
64/128/V/+4/16
AP
64

AP Apple, **Mac** Macintosh, **AT** Atari, **ST**, Atari ST, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **128** Commodore 128, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **AM** Amiga. *General interest

www.commodore.ca

Last month, we hinted at a significant pending announcement for Atari ST users. Here at COMPUTE!, one of the most exciting things we do is launch new publications. We are, without parallel, the most successful and balanced publishing house in the industry of consumer computing. A decision on our part to support a computer manufacturer and a computer system with a dedicated magazine is not made lightly. We are extremely pleased, therefore, to announce that ABC Publishing, our parent company, has committed full support to our launch of COMPUTE!'s Atari ST Disk & Magazine.

This will be our very first product that comes as a magazine/disk combination only. Whether you subscribe or purchase it from a newsstand, you'll get a magazine containing the articles and a disk containing the programs. It's a single, united product. And one we're quite proud of.

No publisher in this industry has been as successful as COMPUTE! Publications at marrying diverse publishing technologies. When we introduced COMPUTE!'s GAZETTE DISK, other disk products were selling a few hundred copies at \$30 or more per issue. We launched the GAZETTE DISK at \$12.95 and created, with your massive support, an overnight price move in the industry. The GAZETTE DISK is today the bestselling product of its kind in the world, circulating tens of thousands of copies per month.

We fully expect COMPUTE!'s Atari ST Disk & Magazine to accomplish the same feat. At launch, our newsstand distribution will rival that of a magazine-only publication. Logistically, there are numerous difficulties involved in binding tens of thousands of disks into magazines heading for newsstands. It's an exciting undertaking, and we'll be anxiously awaiting the results of the first newsstand sales. Watch for the premiere issue of COMPUTE!'s Atari ST Disk & Magazine in September at your local newsstand that handles COMPUTE! and COMPUTE!'s GAZETTE. We have every hope that it will become a collector's item.

You'll find complete details of our announcement on page 3 of this issue. On page 2 you'll also find a rather interesting contest announcement. We're offering \$10,000 in prizes for the very best Atari ST programs and articles. Good luck!

Commodore 64 Forever

At this June's Consumer Electronics Show

in Chicago, Commodore plans to unveil something that may seem ho-hum to many people. In an age of 16/32-bit Amigas and STs with megabytes of memory, Commodore is preparing to announce a revamped version of the Commodore 64—basically the same computer in shiny new wrappings. Dubbed the Commodore 64C, it will be a fully compatible 64 in a Commodore 128-style case. Enclosed in the package will be a floppy disk containing a terminal program for accessing the QuantumLink information service, and GEOS, the graphics-oriented operating system and user interface. Expected price: between \$160 and \$180.

This may not seem too exciting—unless you're a Commodore enthusiast or someone searching for an inexpensive home computer system. From our viewpoint, it's the most exciting 64-related announcement in the past three years. Loud and clear, it broadcasts three important messages:

1. Despite its commitment to establishing the Amiga as its flagship personal computer, Commodore is not abandoning the millions of 64 owners. The Commodore 64C shows that Commodore is determined to continue its support of what has become the world's most popular home computer.

2. The Commodore 64 market will remain a significant source of revenue for software developers, and may even keep expanding.

3. As the bundling of GEOS shows, the 64 is still evolving, growing more powerful and easy to use, and is an exceptional value for people who need a functional computer system for under \$500.

Like Apple's slogan when it introduced the Apple IIc—"Apple II Forever"—Commodore is declaring, in effect, "Commodore 64 Forever."

Forever is a long time, and we don't really think the 64 will be around quite *that* long. Still, Commodore's renewed commitment to the 64 reassures those who have wondered if their computers would soon be "orphans." COMPUTE! has received many letters from readers who feared that the 64 market would dry up and vanish now that Commodore is preoccupied with the Amiga and 128. And actually, as we reported several issues ago, Commodore did attempt to shut down 64 production more than once last year. But each time, the unabated hunger for this four-year-old machine swamped

Commodore with orders, and the company was forced to restart production and rethink its strategy. The 64 refuses to die.

So Commodore is taking advantage of the situation by bringing the computer up-to-date without sacrificing its compatibility with the thousands of programs and peripherals already on the market. Here is what Commodore plans to announce at CES:

- The 64C in a more professional-looking Commodore 128-style case (minus the 128's numeric keypad);

- A bundled disk containing GEOS (Graphics Environment Operating System) and QuantumLink software. GEOS is patterned after the desktops found on the Macintosh, Atari ST, and Amiga—windows, icons, pull-down menus, bit-mapped graphics, and multiple onscreen type fonts. GEOS includes several integrated application programs and desk accessories, including GEOPaint, GEOWrite, a calculator, notepad, and clock. In addition, GEOS significantly speeds up disk access without modifying the 1541 drive. (For more details on GEOS, see our Winter CES report in the April 1986 issue of COMPUTE!.)

- On the flip side of the disk, 64C buyers will get the special terminal software necessary to access QuantumLink, the online communications service specially tailored to Commodore users.

- The 1541 drive will also get a sleek new case to match the 64C.

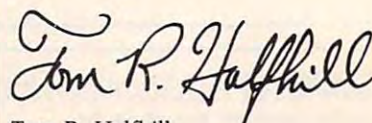
- Memory expansion up to 128K and 512K RAM for the 64 and 128.

- A 3½-inch floppy disk drive for the 64 and 64C, priced around \$225.

All in all, it's an interesting series of announcements, and an encouraging development for Commodore 64 enthusiasts everywhere.



Robert C. Lock,
Editor-In-Chief



Tom R. Halfhill
Editor

IF YOU'RE READY FOR HOME BANKING BUT YOUR BANK ISN'T...

YOU'RE READY FOR CHASE.

NEW YORK'S PREMIER HOME BANKING AND INFORMATION SYSTEM.

You're ready for a way to make the most of your time, your money, and your personal computer, and here it is. SPECTRUMSM, the home banking and information system from The Chase Manhattan Bank, N.A.



BANKING JUST GOT A LOT MORE PERSONAL.

Just switch on your PC, and you've got the undivided attention of a banker, a broker, and a bookkeeper.

Right in your own home, your office, or wherever. 24 hours a day, 7 days a week. With complete security.

IT'S WHAT HOME BANKING SHOULD BE. EASY.

Touch a button to check your balances; transfer funds between checking, savings, money market and other accounts; pay bills to anybody, anywhere; and keep records. Touch another and you've got key financial planning information to set goals...and meet them.

Touch still another and a complete investment menu is at your service: check the market, get the bottom line on over 4600 companies with S&P OnlineSM from Standard & Poor's Corporation, track your portfolio, even

trade stocks and options at a discount through Rose & Company, a Chase affiliate.

If you do have any questions, there's even a real person you can call—16 hours a day.

HOME BANKING, AND THEN SOME.

Since there's more to life than banking and finance, there's more to SPECTRUM than that.

Through our electronic bulletin board, you can find a squash partner, a recipe for pasta al pesto, maybe even a ski house share up in Vermont!

THE BIG PAYOFF.

The good news is that all this starts at just \$5 a month, with the first 2 months of service *free*. There are no sign-up fees or connect time charges, and, thanks to our 800 number, even the phone calls are free.

So if the only thing that's kept you from home banking is your bank, give us a call. We'll make it easy for you to open an account with us to get on-line with SPECTRUM.

You can still use your other bank for other financial needs.

But if you're ready for New York's premier home banking and information system, chances are, you're ready for Chase.

1-800-522-7766.



| | |
|--|--|
| Publisher Founder/Editor in Chief Senior Editor Managing Editor Executive Editor | James A. Casella Robert C. Lock Richard Mansfield Kathleen Martinek Selby Bateman |
| Editor Assistant Editor Production Director Production Editor Editor, COMPUTE!'s GAZETTE Technical Editor Assistant Technical Editor Program Editor Assistant Editor, COMPUTE!'s GAZETTE Assistant Features Editor Programming Supervisor Editorial Programmers Submissions Reviewer Programming Assistants | Tom R. Halfhill Philip Nelson Tony Roberts Gail Cowper Lance Elko Otis R. Cowper George Miller Charles Brannon Todd Heimarck Kathy Yakal Patrick Parrish Tim Victor, Kevin Mykytyn Mark Tuttle David Florance, David Hensley Debi Nash Julia Fleming, Iris Brooks, Mary Hunt, Sybil Agee Jim Butterfield Toronto, Canada Harvey Herman Greensboro, NC Fred D'Ignazio Birmingham, AL David Thornburg Los Altos, CA Bill Wilkinson |
| Executive Assistant Administrative Assistants | |
| Associate Editors | |
| Contributing Editor | |

| | |
|---|--|
| COMPUTE!'s Book Division Editor Assistant Editors Director, Book Sales & Marketing | Stephen Levy Gregg Keizer, Ann Davies Steve Voyatzis |
|---|--|

| | |
|---|---|
| Production Manager Art & Design Director Assistant Editor, Art & Design Mechanical Art Supervisor Artists Typesetting Illustrator | Irma Swain Janice R. Fary Lee Noel De Potter Debbie Bray, Dabney Ketrav Terry Cash, Carole Dunton Harry Blair |
|---|---|

| | |
|---|---|
| Director of Advertising Sales Associate Advertising Director Production Coordinator | Peter Johnsmeyer Bernard J. Theobald, Jr. Kathleen Hanlon |
|---|---|

| | |
|---------------------|---------------|
| Promotion Assistant | Caroline Dark |
|---------------------|---------------|

| | |
|---|--|
| Customer Service Manager Dealer Sales Supervisor Individual Order Supervisor Receptionist Warehouse Manager | Diane Longo Orchid Tamayo Cassandra Green Anita Armfield John Williams |
|---|--|

| | |
|-------------------------|-------------|
| Data Processing Manager | Leon Stokes |
|-------------------------|-------------|

| | |
|---|--|
| James A. Casella, President Richard J. Marino, Vice President, Advertising Sales Chris Savine, Director, Finance & Planning | |
|---|--|

COMPUTE! Publications, Inc. publishes:

COMPUTE!
The Journal for Progressive Computing

COMPUTE!'s GAZETTE
For VIC-20 And Commodore 64 Personal Computers

COMPUTE! Books

COMPUTE!'s GAZETTE DISK

COMPUTE!'s Apple Applications Special

| | |
|--------------------|---|
| Editorial offices: | 324 West Wendover Avenue Suite 200 Greensboro, NC 27408 USA |
| Corporate offices: | 825 7th Avenue New York, NY 10019 212-265-8360 |
| Customer Service: | 800-346-6767 (In NY 212-887-8525) |
| Hours: | 9:30 A.M.-4:30 P.M. Monday-Friday |

Coming In Future Issues

Tightrope:

A Game of Taut Reflexes For
Commodore 64/128, Atari, Apple,
Amiga, And IBM PC/PCjr

Sprite-32 For Commodore 64:
How To Display 16 Or 32 Sprites

Softball Statistics For Atari ST

The Logical Alternative For Atari
400/800/XL/XE:

Replace Slow IF-THENS With Fast Logic

Apple ProDOS Protector And DOS 3.3
Guardian Angel:
Protect Your Programs From
Prying Eyes

Commodore 128 Machine Language:
A New Series By Jim Butterfield
Foolproof Input For Amiga BASIC

Subscription Orders

COMPUTE!
P.O. Box 10954
Des Moines, IA 50340

TOLL FREE
Subscription Order Line
800-247-5470
In IA 800-532-1272

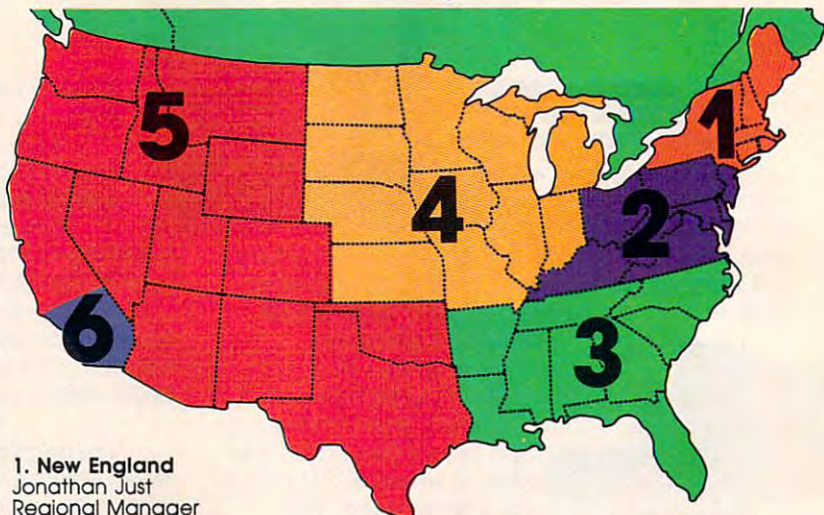
COMPUTE!
Subscription Rates
(12 Issue Year):

US (one yr.) \$24
(two yrs.) \$45
(three yrs.) \$65
Canada and Foreign
Surface Mail \$30
Foreign Air
Delivery \$65



Magazine Publishers Association

Advertising Sales



1. New England
Jonathan Just
Regional Manager
212-315-1665

2. Mid Atlantic
Jonathan Just
Regional Manager
212-315-1665

3. Southeast & Foreign
Harry Blair
919-275-9809

4. Midwest
Gordon Benson
312-362-1821

5. Northwest/Mountain/Texas
Phoebe Thompson
Dani Nunes
408-354-5553

6. Southwest
Ed Winchell
213-378-8361

Director of Advertising Sales:
Peter Johnsmeyer

Associate Advertising Director:
Bernard J. Theobald, Jr.

COMPUTE! Home Office 212-887-8460.

Address all advertising materials to:
Kathleen Hanlon
Advertising Production Coordinator
COMPUTE! Magazine
324 West Wendover Avenue
Suite 200
Greensboro, NC 27408

The COMPUTE! subscriber list is made available to carefully screened organizations with a product or service which may be of interest to our readers. If you prefer not to receive such mailings, please send an exact copy of your subscription label to: COMPUTE! P.O. Box 10955, Des Moines, IA 50340. Include a note indicating your preference to receive only your subscription.

Authors of manuscripts warrant that all materials submitted to COMPUTE! are original materials with full ownership rights resident in said authors. By submitting articles to COMPUTE!, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of COMPUTE! Publications, Inc. No portion of this magazine may be reproduced in any form without written permission from the publisher. Entire contents copyright © 1986, COMPUTE! Publications, Inc. Rights to programs developed and submitted by authors are explained in our author contract. Unsolicited materials not accepted for publication in COMPUTE! will be returned if author provides a self-addressed, stamped envelope. Programs (on tape or disk) must accompany each submission. Printed listings are optional, but helpful. Articles should be furnished as typed copy (upper- and lowercase, please) with double spacing. Each page of your article should bear the title of the article, date and name of the author. COMPUTE! assumes no liability for errors in articles or advertisements. Opinions expressed by authors are not necessarily those of COMPUTE!.

PET, CBM, VIC-20 and Commodore 64 are trademarks of Commodore Business Machines, Inc. and/or Commodore Electronics Limited. Apple is a trademark of Apple Computer Company. IBM PC and PCjr are trademarks of International Business Machines, Inc.

ATARI is a trademark of Atari, Inc. TI-99/4A is a trademark of Texas Instruments, Inc. Radio Shack Color Computer is a trademark of Tandy, Inc.

Color Monitor



(Premium Quality)

- Built in Speaker & Audio
- For Video Recorders
- For Small Business Computers
- Apple - Commodore - Atari - Aplus 3000 -etc.
- One Year Warranty'

Sale



(Premium Quality)

- Beautiful Color Contrast
- High Resolution
- Sharp Clear Text
- Anti-Glare Screen
- 40 Columns x 24 Lines
- Front Panel Controls

List \$329⁰⁰

Sale \$139⁹⁵*

Add \$14.50 Shipping



13" Color Computer Monitor'

*C64/Atari composite cable \$9.95

* C128 RGB/Composite 80 column cable \$19.95.

14" RGB & Composite Color Monitor

Allows use of C-128 and C64 mode - composite and 80 column RGB mode. Must be used to get 80 columns in color with 80 column computers. Specially designed for use with the C128's special composite video output, plus green screen only option switch. (add \$14.50 shipping)

List \$399.00

Sale \$259⁹⁵*

14" MAGNAVOX Higher Resolution RGB & Composite Monitor

(Add \$14.50 Shipping)

Sale \$279⁹⁵*

12" 80 Column Green/Amber Monitor

List \$129.00

Super high resolution composite green or amber screen monitor. 80 columns x 24 lines, easy to read. Fantastic value. Limited Quantities.

Sale \$79⁹⁵*

9" Samsung Hi Res Green Screen Monitor

Super High Resolution 80 column monitor perfect for Apple & Aplus 3000 computers. Fantastic Value. Very Limited Quantities.

List \$129.95

Sale \$59⁹⁵*

Turn Your Monitor into a TV Set Without Moving Your Computer

Elegant TV Tuner with dual UHF/VHF selector switches goes between your computer and monitor. Includes mute, automatic fine tuning and computer-TV selector switches. Inputs included for 300 ohm, 75 ohm, and UHF. Can be used with cable TV and VCR's. Fantastic Value. Limited Quantities. (Includes loop antenna for UHF & RCA connecting cables)

List \$129.95

Sale \$49⁹⁵

15 Day Free Trial - 90 Day Immediate Replacement Warranty

• LOWEST PRICES • BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL • OVER 500 PROGRAMS • FREE CATALOGS

Add \$10.00 for shipping, handling and insurance. Illinois residents please add 6 1/4% tax. Add \$20.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES, EXCEPT CANADA. Enclose Cashier Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders, 1 day express mail! Prices & Availability subject to change without notice. VISA — MASTER CARD — C.O.D. C.O.D. on phone orders only

PROTECTO

We Love Our Customers

22292 N. Pepper Rd., Barrington, Illinois 60010

312/382-5244 to order

COMMODORE 64 COMPUTER

(Order Now)

\$139.95

- C128 Disks 79¢ ea.*
- Paperback Writer 64 \$39.95
- 13" Color Monitor \$139.95

CALL BEFORE YOU ORDER

COMMODORE 64 SYSTEM SALE

Commodore 64 Plus \$30.00 S&H

Com. 1541
Disk Drive

13" Color
Monitor

\$457

C128 COMMODORE COMPUTER

(Order Now)

* \$229.05

(SEE BELOW)

With \$59.95 Timeworks Wordwriter
Wordprocessor savings applied

- 340K 1571 Disk Drive \$259.00
- Voice Synthesizer \$39.95
- 12" Monitor \$79.95

PRICES MAY BE LOWER

SPECIAL SOFTWARE COUPON

We pack a SPECIAL SOFTWARE DISCOUNT COUPON with every COMMODORE 64 COMPUTER, DISK DRIVE, PRINTER, or MONITOR we sell! This coupon allows you to SAVE OVER \$250 OFF SALE PRICES!!

(Examples)

PROFESSIONAL SOFTWARE COMMODORE 64

| Name | List | Sale | Coupon |
|---------------------------------|---------|---------|---------|
| PaperClip | \$59.95 | \$34.95 | \$29.95 |
| Consultant | \$59.95 | \$49.95 | \$39.95 |
| Leader Board | \$39.95 | \$24.95 | \$22.95 |
| The Print Shop | \$44.95 | \$27.95 | \$26.95 |
| Halley's Project | \$39.95 | \$22.95 | \$19.95 |
| Practicalc (spread sheet) | \$59.95 | \$19.95 | \$14.95 |
| Voice Command Module | \$79.95 | \$39.95 | \$34.95 |
| Nine Princes in Amber | \$32.95 | \$24.95 | \$21.95 |
| Super Bowl Sunday | \$35.00 | \$22.95 | \$19.95 |
| Flip and File Disk Filer | \$24.95 | \$14.95 | \$12.95 |
| Pro Joy Stick | \$19.95 | \$12.95 | \$10.00 |
| PartyWare | \$19.95 | \$14.95 | \$11.95 |
| Dust Cover | \$ 8.95 | \$ 6.95 | \$ 4.60 |
| Financial Planner | | | |
| Sylvia Porter | \$59.95 | \$38.95 | \$35.95 |
| Hardball | \$29.95 | \$18.95 | \$16.95 |
| C64 Troubleshoot & Repair Guide | \$24.95 | \$15.95 | \$12.95 |

(See over 100 coupon items in our catalog)

Write or call for
Sample SPECIAL SOFTWARE COUPON!

ATTENTION Computer Clubs We Offer Big Volume Discounts CALL TODAY!

PROTECTO WARRANTY

All Protecto's products carry a minimum 90 day warranty. If anything fails within 90 days from the date of purchase, simply send your product to us via United Parcel Service prepaid. We will IMMEDIATELY send you a replacement at no charge via United Parcel Service prepaid. This warranty proves once again that **We Love Our Customers.**

COMMODORE 64 COMPUTER \$139.95

You pay only \$139.95 when you order the powerful 84K COMMODORE 64 COMPUTER! LESS the value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your computer that allows you to SAVE OVER \$250 off software sale prices!! With only \$100 of savings applied, your net computer cost is \$39.95!!

* C128 DOUBLE SIDED DISKS 79¢ EA.

Get these 5 1/4" Double Sided Floppy Disks specially designed for the Commodore 128 Computer (1571 Disk Drive). 100% Certified, Lifetime Warranty, Automatic Lint Cleaning Liner included. 1 Box of 10 - \$9.90 (99¢ ea.), 5 Boxes of 10 - \$44.50 (89¢ ea.), 10 Boxes of 10 - \$79.00 (79¢ ea.).

13" COLOR MONITOR \$139.95

You pay only \$139.95 when you order this 13" COLOR MONITOR. LESS the value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your monitor that allows you to save over \$250 off software sale prices!! With only \$100 of savings applied, your net color monitor cost is only \$39.95. (16 Colors).

Premium Quality 120-140 CPS Comstar 10X Printer \$148.00

The COMSTAR 10X carriage, 120-140 CPS, 9 double strike capability (near letter quality), 120 x 144 dot matrix, left and right margin set with super quality, block graphics and round on printers costing twice as much!! (Centronics Parallel Interface) List \$399.00 Sale \$148.00.

4 SLOT EXPANDER & 80 COLUMN BOARD \$49.95

Now you program 80 COLUMNS on the screen at one time! Converts your Commodore 64 to 80 COLUMNS when you plug in the 80 COLUMN EXPANSION BOARD!! PLUS 4 slot expander! Limited Quantities. Sale \$49.95. Coupon \$39.95

80 COLUMNS IN COLOR

PAPERBOCK WRITER 64 WORD PROCESSOR \$29.95

This PAPERBOCK WRITER 64 WORD PROCESSOR is the finest available for the COMMODORE 64 computer! The ULTIMATE FOR PROFESSIONAL Word Processing. DISPLAYS 40 or 80 COLUMNS IN COLOR or black and white! Simple to operate, powerful text editing, complete cursor and insert/delete key controls line and paragraph insertion, automatic deletion, centering, margin settings and output to all printers! List \$99.00. SALE \$29.95.

* C128 COMMODORE COMPUTER \$289.00

You pay only \$289.00 for the C128 computer and we include the C128 Wordwriter Wordprocessor by Timeworks (Sale \$59.95). Thus, your net cost for the C128 computer is only \$229.05. List \$349.00. SALE \$289.00.

340K 1571 COMMODORE DISK DRIVE \$259.00

Double Sided, Single Disk Drive for C-128 allows you to use C-128 mode plus CPM mode. 17 times faster than 1541, plus runs all 1541 formats. List \$349.00. Sale \$259.00.

SUPER AUTO DIAL MODEM \$29.95

Easy to use. Just plug into your Commodore 64 computer and you're ready to transmit and receive messages. Easier to use than dialing your telephone, just push one key on your computer! Includes exclusive easy to use program for up and down loading to printer and disk drives. Best In U.S.A. List \$99.00. SALE \$29.95. Coupon \$24.95.

VOICE SYNTHESIZER \$39.95

For Commodore-64 computers. Just plug it in and you can program words and sentences, adjust volume and pitch, make talking adventure games, sound action games and customized talkies!! PLUS (\$19.95 value) TEXT TO SPEECH program included FREE, just type a word and hear your computer talk — ADD SOUND TO "ZORK", SCOTT ADAMS AND OTHER ADVENTURE GAMES!! (Disk or tape.) List \$89.00. SALE \$39.95

12" MAGNAVOX (NAP) 80 COLUMN MONITOR WITH SOUND \$79.95

Super High Resolution green screen monitor. 80 columns x 24 lines, easy to read, plus speaker for audio sound included. Fantastic value. List \$129.00. Sale \$79.95. (C128 cable \$19.95. C64, Atari cable \$9.95)

PRINTER/TYPEWRITER COMBINATION \$229.95

"JUKI" Superb letter quality, daisy wheel printer/typewriter combination. Two machines in one — just a flick of the switch. 12" extra large carriage, typewriter keyboard, automatic margin control and relocate key, drop in cassette ribbon! (90 day warranty) centronics parallel or RS232 serial port built in (Specify). List \$349.00. SALE \$229.95. (Ltd. Qty.)

14" RGB & COMPOSITE COLOR MONITOR \$259.95

Must be used to get 80 columns in color with 80 column computers (C128 - IBM - Apple). (RGB Cable \$19.95) Add \$14.50 shipping. List \$399.00. SALE \$259.95.

- LOWEST PRICES • 15 DAY FREE TRIAL
- BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL

PHONE ORDERS

8 a.m. - 8 p.m. C.S.T. Weekdays
9 a.m. - 12 noon C.S.T. Saturdays

- 90 DAY FREE REPLACEMENT WARRANTY
- OVER 500 PROGRAMS • FREE CATALOGS

Add \$10.00 for shipping, handling and insurance. Illinois residents please add 6 1/4% tax. Add \$20.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES, EXCEPT CANADA. Enclose Cashier Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders, 1 day express mail! Prices & Availability subject to change without notice. VISA — MASTER CARD — C.O.D. No. C.O.D. to Canada, APO-FPO

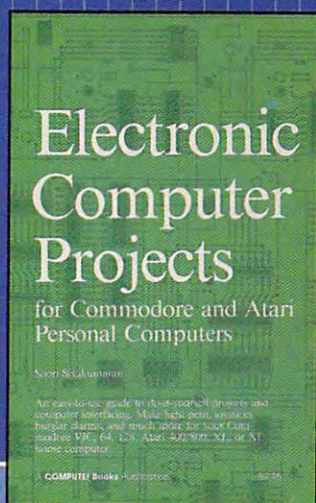
PROTECTO

We Love Our Customers
22292 N. Pepper Rd., Barrington, Illinois 60010
312/382-5244 to order

www.commodore.ca

Two Exciting New Books

from
COMPUTE!



COMPUTE!'s First Book of the Commodore 128

A spectacular collection of articles and programs exclusively for the Commodore 128 in 128 mode.

Edited

The editors at COMPUTE! Publications have collected some of the best games, programs, and tutorials for the Commodore 128 from *COMPUTE!* and *COMPUTE!'s Gazette*, plus some never-before-published articles and programs. Learn how to create windows, program sound, and make disks autoloading. You'll even find a map of all the important memory locations. There's something for every 128 user. All programs run in 128 mode. *A disk is available which includes programs in the book, \$12.95.*

\$14.95 ISBN 0-87455-059-9

Electronic Computer Projects

Learn how to build all kinds of new devices to interface with your computer from inexpensive, available parts.

For the Commodore 64, 128, VIC, and any eight-bit Atari personal computer.

Soori Sivakumaran

This introduction to digital electronics and computer interfacing is the easy way to learn how computers interact with the outside world. Using a Commodore 64, 128, VIC, or any eight-bit Atari computer and *Electronic Computer Projects*, you'll be guided through the steps to building a joystick, light pen, game paddle, and numerous other devices. And since each project is independent from the others, you can choose only those projects that interest you. All the projects can be built at home and most require fewer than half a dozen parts.

\$9.95 ISBN 0-87455-052-1

Visit your local book or computer store for these new titles. Or order directly from COMPUTE! Books. Call toll-free 800-346-6767 (in NY 212-887-8525) or write COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

Please include \$2.00 per book (\$5.00 air mail) for shipping and handling. NC residents add 4.5 percent sales tax. Allow 4-6 weeks from receipt of order for delivery.

COMPUTE! Publications, Inc. abc

Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
825 7th Avenue, 6th Floor, New York, NY 10019
Publishers of COMPUTE!, COMPUTE!'s Gazette, COMPUTE!'s Gazette Disk, COMPUTE! Books, and COMPUTE!'s Apple Applications

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England, and in Canada from McGraw-Hill, Ryerson Ltd., 330 Progress Ave., Scarborough, Ontario, Canada M1P 2Z5.

 www.commodore.ca



Readers Feedback

The Editors and Readers of COMPUTE!

If you have any questions, comments, or suggestions you would like to see addressed in this column, write to "Readers' Feedback," COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Due to the volume of mail we receive, we regret that we cannot provide personal answers to technical questions.

The Ideal BASIC Style

Some time ago I read a letter in your magazine regarding crunching of program listings and the effect this has on readability. You replied that this was to save memory and magazine space. I would like to suggest a reasonable compromise between readability and the elimination of spaces. In my view, any statement that juxtaposes two letters (for instance, `FOR T=1 TO 10` or `IF S=5`) would benefit greatly from extra spaces (`FOR T=1 TO 10` or `IF S=5`). But if a number follows a letter (as in `GOTO600` or `THEN470`), the statement is understandable even without an extra space. I think `DATA` should always have a following space so the first value stands out clearly. You are often inconsistent in this, even within the same program listing. As to multiple statements in one line, this in itself creates no problems and is necessary in some cases. But I don't believe that completely unrelated statements should be put on the same line simply to fill up the line.

Line numbering is another area. You use the time-honored decade numbering (line increments of ten), which is fine when developing a program. Finished programs are usually renumbered for neatness, but I don't see why nine skipped numbers are necessary. I suggest that you use every other number instead (1, 3, 5, and so on). This would allow someone to insert a `STOP` or `GOTO` while checking for typing errors or making minor alterations. The big advantage of this system is that decade numbers could have special meanings as important entry points or the beginning of a new group of closely related statements. For example, a complex `FOR-NEXT` loop might use several lines, then jump to the next decade line number for a new group of related

statements. It would be much easier to follow and understand the flow pattern.

I also feel there could be at least partial standardization of some of the most common variable names. For instance, the variables `I`, `J`, and `K` are ordinarily used as "junk" variables (counters within loops, and so on). The variables `X` and `Y` are frequently used to specify horizontal and vertical coordinates. But many others are commonly used as well: `SA` for starting address, `EA` for ending address, `CK` for checksum, and so forth. You could publish a list of suggested variable names and encourage programmers to stick to it.

Don R. King

As long as programmers use BASIC, there will be discussions about what sort of style and structure BASIC programs ought to have. The reason for the controversy is familiar. BASIC imposes few structural constraints on the programmer, so the language is easy to learn and works well for improvisational programming and quick experiments. But its lack of structure also makes it possible to write tangled, illogical "spaghetti" code. Since BASIC doesn't force you into a predetermined mold, a program can take nearly any form. More structured programming languages such as Pascal generate more readable code, but demand more forethought on the programmer's part.

Most of the programs we publish are submissions from readers. Generally, we modify these programs only to eliminate any bugs that appear during testing or to add functional improvements. Any time you change someone else's program, you increase the likelihood of inadvertently creating new bugs which even the author may not have anticipated. Given the number of programs we publish and the constraints of monthly deadlines, it's not practical for us to rewrite working programs merely to improve their readability.

A carefully planned numbering scheme can add to a program's readability. But our programs are meant to be typed in from a printed listing as well as studied. So we need to do everything possible to help readers type the programs without errors. Numbering in regular increments makes it easier to keep your place in the program than if the increments changed at unpredictable intervals. Uniform num-

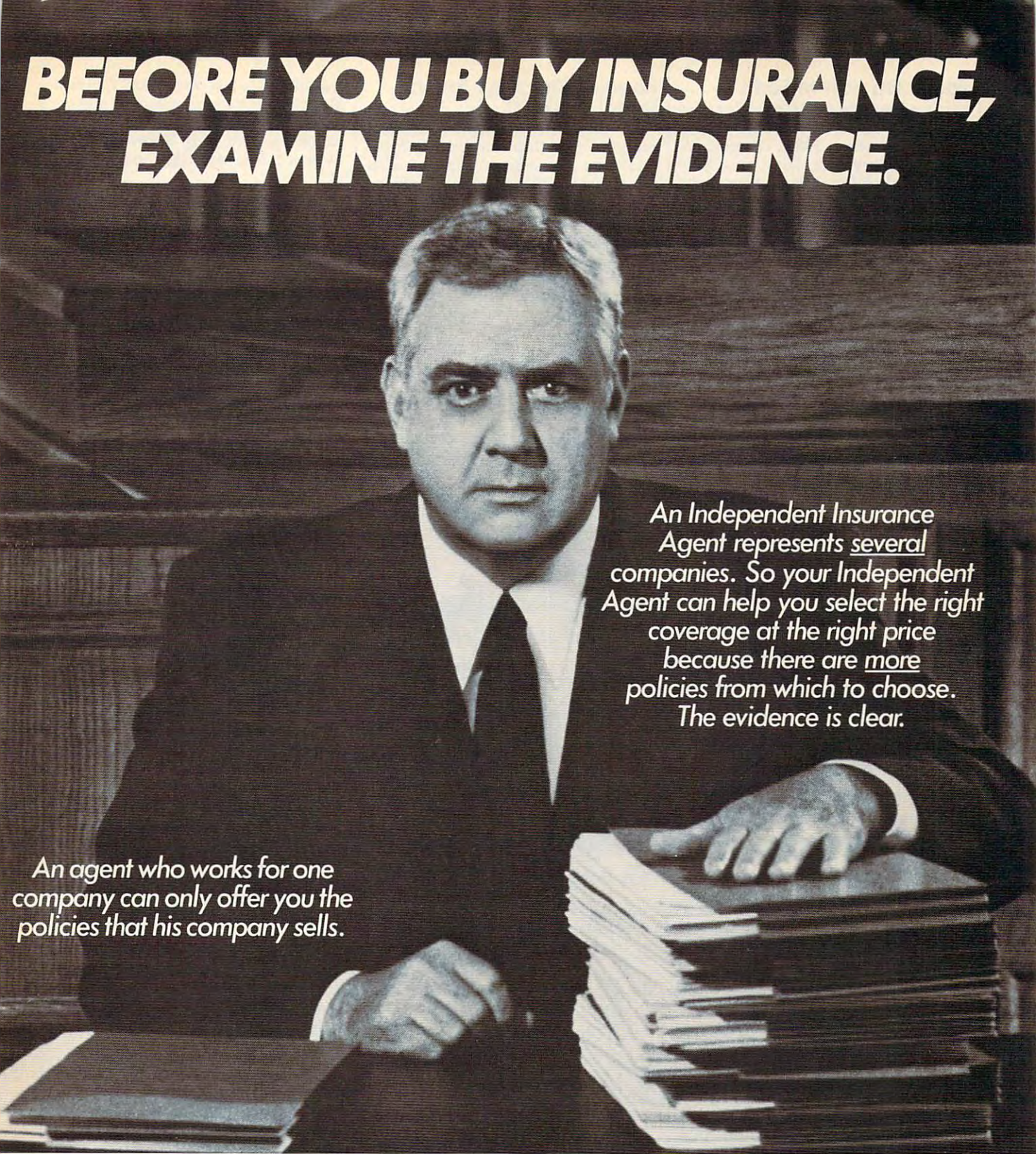
bering also helps readers spot lines that have been left out altogether (a typing error that no proofreader program can catch). However, sometimes even the simple act of renumbering a program can introduce new bugs—as has happened to us in the past.

It's also true that if everyone followed the same stylistic conventions, BASIC programs would be more readable. The difficult part is getting programmers to go along with the conventions you choose, especially considering that each version of BASIC has its own peculiarities. For instance, Commodore BASIC doesn't require spaces after keywords (and omitting spaces speeds program execution), but some other versions of Microsoft BASIC insist on a separating space. Other BASICs, such as Atari BASIC, automatically insert spaces for readability if you leave them out.

Different dialects of BASIC also include different keywords. For instance, `NAME` is a legal variable name in Commodore BASIC, but it's treated as a reserved word in IBM BASIC and Amiga BASIC. In Commodore and Apple BASICs, only the first two characters of the variable name are significant, and you may not embed keywords in variable names. But IBM, Atari, and Amiga BASICs permit long, descriptive variable names such as `MousePosition` or `MenuFlag` which can include embedded keywords. The list of differences goes on and on. Given the diversity among BASIC dialects and the absence of standardization, any list of preferred variable names would have to be exceedingly general and geared toward the lowest common denominator.

As time goes by, Microsoft BASIC seems to be taking over as the de facto standard for the language. Newer, more powerful computers such as the Macintosh, Atari ST, and Amiga all offer versions of BASIC that more closely resemble IBM BASIC. With the exception of graphics and sound statements, which are necessarily hardware-specific, a program that runs on the IBM, Mac, or Amiga will probably run on any of the others with only slight modifications. If this trend continues, we may someday reach the point where BASIC style becomes more homogeneous.

BEFORE YOU BUY INSURANCE, EXAMINE THE EVIDENCE.



An Independent Insurance Agent represents several companies. So your Independent Agent can help you select the right coverage at the right price because there are more policies from which to choose. The evidence is clear.

An agent who works for one company can only offer you the policies that his company sells.

THE MORE-THAN-ONE-COMPANY INSURANCE AGENT.

You'll find the Independent Insurance Agent nearest you in the Yellow Pages.



Faster Fractals In Forth

I enjoyed reading Paul Carlson's article on fractal graphics for the IBM PC/PCjr (COMPUTE!, March 1986). His explanations were very clear. But it must have been a real trial for him to develop the BASIC version of the "Eight Thousand Dragons" program. We would like to show the beauty of fractals when written with a language that supports recursion. Here is an example of Forth code that does the same thing. It's written for Mach1, our Forth compiler for the Apple Macintosh and Atari ST. The execution time for a fourteenth-degree dragon is only three minutes.

Reading The Atari Touch Tablet In BASIC

I am currently working on an Atari program that lets me create high-resolution drawings in graphics mode 15. However, the drawing should be done with the Atari Touch Tablet. How can a program read the Touch Tablet coordinates?

Peter Hinz

Reading coordinates from the Atari Touch Tablet is very easy in Atari BASIC. The Touch Tablet returns the same values as paddle controllers, and Atari BASIC contains a function called PADDLE for reading these controllers. Use PADDLE(0) to

read the left button, and PTRIG(1) to read the right button (again, assuming that the tablet is plugged into port 1). When a button is pressed, these functions return a value of 0. Otherwise, they return a value of 1.

The button on the Touch Tablet's stylus works a little differently. To detect this button press, use the STICK(0) function (normally intended for reading a joystick). If the stylus button is pressed, STICK(0) returns a value of 14. Otherwise, it returns the value 15.

The following example program prints the tablet coordinates on the screen along with messages when any of the buttons are pressed:

```
BN 10 X=PADDLE(0):Y=PADDLE(1)
MM 20 PRINT X,Y
ID 30 IF PTRIG(0)=0 THEN PRINT "LEFT BUTTON PRESSED"
NI 40 IF PTRIG(1)=0 THEN PRINT "RIGHT BUTTON PRESSED"
HL 50 IF STICK(0)=14 THEN PRINT "STYLUS BUTTON PRESSED"
AA 60 GOTO 10
```

Safe Zones In IBM BASIC

Is there any way to store a few characters or flags in the IBM PC's memory that will survive the BASIC RUN command? I want my program to be able to "learn" as it runs and remember what it has learned each time it is run.

H. Beck

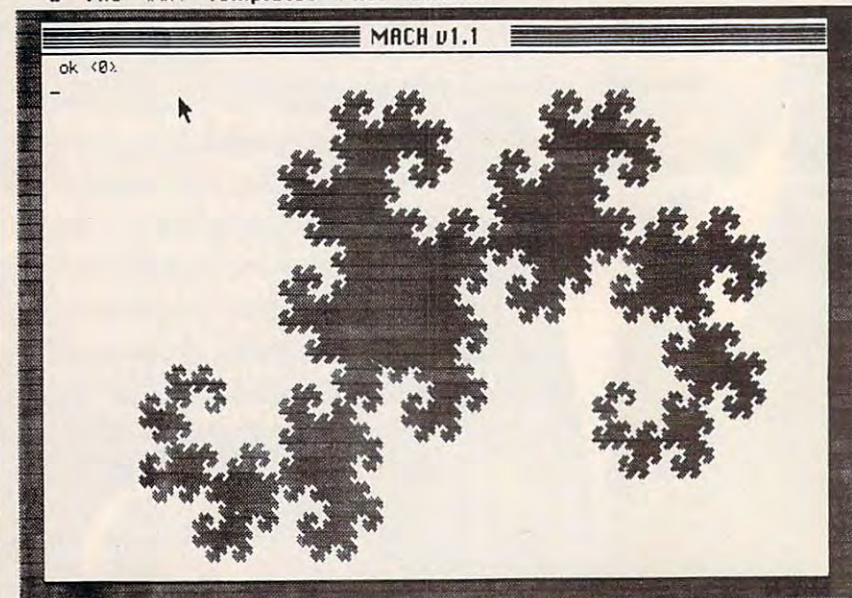
IBM BASIC's CLEAR command gives you the ability to create a safe area of RAM of almost any size. Besides deleting all variables, CLEAR controls the amount of memory available to BASIC. By adding a comma and a parameter to the CLEAR command, you can make the BASIC workspace smaller than usual, reserving the extra memory for yourself. The workspace is initially 65,536 bytes, but it's easy to reserve some memory at the top of that space. Use this format:

CLEAR ,workspace

where workspace is a number less than 65536. To calculate the correct value, subtract from 65536 the number of bytes you want to protect. For instance, the command CLEAR ,65280 reserves the last 256 bytes (65536 - 256 = 65280) of BASIC workspace for your use.

When you type RUN after a CLEAR statement like this, the size of the workspace is reset to its default but the data in the reserved area is not affected. As long as the next program begins with a similar CLEAR statement, it can PEEK into the reserved area and find the values that the previous program POKed there. Here's a

File Edit Templates Misc Utilities



```
: Drag RECURSIVE { x1 y1 x2 y2 x3 y3 k | x4 y4 x5 y5 }
  PAUSE
  k 0= (if k=0 just draw and return)
  IF (else continue breaking lines down)
    k 1- -> x4
    x1 x2 + 2/ y2 y1 - 2/ - -> x4
    y1 y2 + 2/ x2 x1 - 2/ + -> y4
    x2 x3 + 2/ y3 y2 - 2/ + -> x5
    y2 y3 + 2/ x3 x2 - 2/ - -> y5
    x1 y1 x4 y4 x2 y2 k Drag
    x2 y2 x5 y5 x3 y3 k Drag
  THEN ;
:Dragon { iters --} ('14 dragon' gives the best results)
  CLS
  100 190 CALL MoveTo (place pen at beginning)
  100 190 228 62 356 190 iters Drag (start with initial seed)
```

Terry Noyes

Although recursive routines (program segments that call themselves) are ordinarily taboo in BASIC, they're not only feasible, but encouraged in other languages such as Logo and Forth. Besides speeding execution, recursion produces compact, elegant code, as this example shows. Thanks for the demonstration.

read the horizontal position of the stylus on the tablet, and PADDLE(1) to read the vertical position (assuming that the tablet is plugged into controller port 1). Both functions return values ranging from 1 to 228. When nothing is touching the tablet surface, these functions return the value 228.

Reading the Touch Tablet buttons is just as easy. Use the PTRIG(0) function to

simple program that stores some values in a 256-byte reserved area:

```
10 CLEAR,65280
20 FOR A=0 TO 255
30 POKE A+65280,A
40 NEXT
```

After you run the program, enter and run this program to read the stored values back.

```
10 CLEAR,65280
20 FOR A=0 TO 255
30 PRINT PEEK (A+65280)
40 NEXT
```

Scanning The 128's ALT Key

Please tell me how to read the ALT key on the Commodore 128.

J. C. Vollmer

The 128's ALT key cannot be polled like the other keys. Instead, your program must PEEK location 211, where the system stores information about five special keys: SHIFT, CONTROL, ALT, CAPS LOCK, and the Commodore key. When you press one of these keys, it sets a certain bit in this location:

| Key | Bit | PEEK(211) Value |
|-----------|-----|-----------------|
| SHIFT | 0 | 1 |
| Commodore | 1 | 2 |
| CONTROL | 2 | 4 |
| ALT | 3 | 8 |
| CAPS LOCK | 4 | 16 |

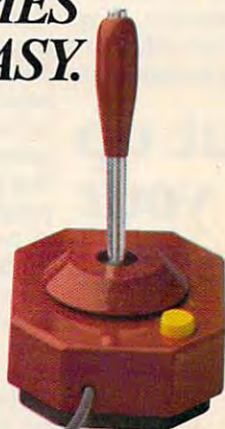
Thus, if PEEK(211) equals 8, you know the ALT key is pressed. Since each key has its own signal bit, the values from location 211 are additive. If PEEK(211) equals 9, for instance, both SHIFT and ALT are pressed. When SHIFT and CONTROL are pressed, location 211 holds 5, and so forth.

Correction For Casio Review

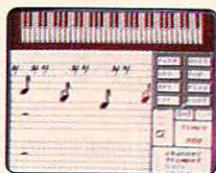
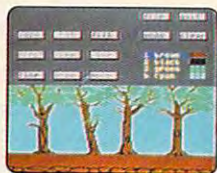
I enjoyed reading your January 1986 review of *The Music Shop for MIDI* and the Casio CZ-101 synthesizer. In fact, I became inspired and bought the same system, after having exhausted the 64's musical capabilities. You mentioned a problem with accessing all the features of the CZ-101 synthesizer. Perhaps I obtained an updated version of *The Music Shop for MIDI* because I have not had the same experience. All 48 timbres can be accessed (the basic presets plus those in internal memory or cartridge memory). In addition, four-voice polyphonic music is possible within the program using the programmable MIDI features (channeling the four solo voices). Casio's COSMO series of synthesizers, which includes the CZ-101, is capable of playing up to eight timbres (four on the CZ-101) on a single slave unit. Have you looked into other types of MIDI software presently available?

DESIGNING YOUR OWN COMPUTER GAMES JUST BECAME EASY.

Have a great idea for a game? Don't have enough time to learn how to turn it into software? Your magic wand has just arrived. Activision proudly presents Garry Kitchen's GameMaker: The Computer Game Design Kit.™ We've packed five professional-quality design tools into one easy-to-use program.



SceneMaker: Design the set. Select from preprogrammed backgrounds like space, jungle or river scenes or create a world of your own.



MusicMaker: Compose the score. Set the mood with just the right music or create triumphant interludes.



SoundMaker: What do you get when you cross a "clunk!" with a "boom?" From explosions to train whistles—smash, blast and whoosh your way into a smorgasbord of sound effects.



SpriteMaker: Who's who and what's what. Create and animate the characters and objects that move across the screen.



The Editor: The grand finale. Look at all the components, choose some and edit others, polish it and ... bring it to life. We've even given you a blank disk so you can send it to a friend ... or publisher.



For the Apple II series, Commodore 64 or 128 and compatible computers.

ACTIVISION
CREATIVITY SOFTWARE®

GameMaker: Unleash the power of your computer—and your imagination. Then, revel in the creation of a true original ... a game of your own.

For the dealer nearest you call (800) 227-9759 (in California, (415) 960-0410 weekdays only). Apple is a trademark of Apple Computer. Commodore is a trademark of Commodore Electronics Limited. Activision is the registered trademark of Activision, Inc. © 1986 Activision, Inc. P.O. Box 7287, Mountain View, CA 94039

And your Earls and Viscounts. If you've got royal ancestors, we have the noble software that can help you trace them down.

Family Roots and your Apple, IBM, Commodore, Kaypro*, and many others, offer individual and group sheets, charts, name indices, general search and text

Put up your dukes!

capabilities. Adapts to most disk drives, printers, and screens. You get more utility programs, plus lots of personal control. A comprehensive (new) manual is included.

All for just \$185.

Write or call today for more information and a free brochure.

Quinsept, Inc.

P.O. Box 216
Lexington, MA 02173
(617) 641-2930

American Express, Visa, and MasterCard gladly accepted.

*Trademarks for Apple Computer Inc., International Business Machines, IBM, and Digital Research.



EMERGENCY POWER SYSTEM

FULL Back-Up Computer Protection!

as low as
\$359



Transfer time to emergency power 10 Milliseconds. Self-contained with enclosed gel cell battery. 425-Watt and 200-Watt 28 ampere models operate up to 35 minutes allowing ample time for safe shutdown! 3-Way AC line filter stops transient spikes and surges. 4 Receptacles. Automatic regulated battery charger. Output voltage 117VAC, 60 hz. frequency controlled $\pm 1/2$ cycle.

- ☐ 200-Watt (10 ampere hours) only \$359
- ☐ 200-Watt (28 ampere hours) only \$429
- ☐ 425-Watt (28 ampere hours) only \$599

Order toll free 1-800-662-5021

IN ILLINOIS, CALL 1-312-648-2191 OR MAIL COUPON

INDUS-TOOL, 730 W. Lake Street
Dept. CI, Chicago, IL 60606

Enclosed is \$ _____ or charge on

☐ MasterCard or ☐ Visa Expires _____

Card no. _____

Send model # _____

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Please continue your articles on computer music.

Eric Habeck

Thanks to you and to Don Williams (the programmer who created The Music Shop and The Music Shop for MIDI) for alerting us to the misstatement. The 64 is very popular with musicians and sound enthusiasts because of its low cost and built-in sound capabilities. As the MIDI standard becomes more widely accepted, we're likely to see even more in the way of music software for the 64. We'll continue to review new products as time and space permit.

Customizing SpeedScript 3.0

The "SpeedScript Customizer" program that appeared in the September 1984 issue of COMPUTE's GAZETTE allows you to change the default settings and formatting commands in SpeedScript 1.0 to fit your own preferences. But that program doesn't work with SpeedScript 3.0 or 3.2. Do you have an update of the program or the necessary POKES to allow the same customization for the most recent versions of SpeedScript?

Bruce Patten

It just so happens that another reader wrote in with the very information you're seeking:

"SpeedScript Customizer" doesn't work with SpeedScript 3.0 or 3.2. But I have discovered the POKES for customizing all of the same parameters for the newest versions of SpeedScript. Here are the default (normal) values and the locations that control them:

| Default Location | Value | Parameter |
|------------------|-------|-----------------------------------|
| 5722 | 5 | Left margin |
| 5723 | 75 | Right margin |
| 5724 | 66 | Page length |
| 5725 | 5 | Top margin |
| 5726 | 58 | Bottom margin |
| 5727 | 1 | Spacing |
| 5728 | 1 | Wait (1=go ahead) |
| 5729 | 1 | @ start numbering pages at (LSB) |
| 5730 | 0 | (MSB) |
| 5731 | 1 | ? starting printing at page (LSB) |
| 5732 | 0 | (MSB) |
| 5733 | 80 | x columns across |
| 5734 | 27 | 1 printkey 1 |
| 5735 | 14 | 2 printkey 2 |
| 5736 | 15 | 3 printkey 3 |
| 5737 | 18 | 4 printkey 4 |

To customize your program, load SpeedScript 3.0 or 3.2 into memory, then POKE the desired values into the appropriate memory locations. For example, POKE 5722,3 makes the left margin setting default to 3 instead of 5. Then save the program using a different name. For instance, I have a frequent

need to print postcards, so I set the left margin at 3, right margin at 35, columns at 40, top margin at 3, bottom margin at 18, and page length at 2. If you want to start numbering pages or start printing pages at a page lower than 256, POKE the desired value in the first of the two locations indicated. For instance, to start numbering pages at page 3, you would POKE 5729,3. To start at a page higher than 255, you must POKE two values in low byte/high byte format. The low byte of the value goes into the lower location.

Allen Perkins

Thank you for the information. As mentioned in the original SpeedScript article, most of these settings have to do with formatting hardcopy printouts.

Apple RESET Vectoring

Is there any way to make the Apple II jump to a specific machine language subroutine after the RESET key has been hit?

Jose A. Colon Olivo

This can be done by changing the two-byte RESET vector at location 1010 (\$03F2). The most direct way to alter the vector is to POKE the starting address of your machine language program into locations 1010-1011 in low byte/high byte format, then update these pointers with CALL -1169. When you hit RESET, the Apple checks the vector, goes to the indicated location, and runs your program. As an example, suppose you wish to execute the following routine which prints an A on the screen upon RESET:

```
0300 LDA #C1
0302 JSR $FDF0
0305 JMP $03D0
```

The first step is to determine the high and low bytes of the starting address in decimal. The hexadecimal number \$0300 is expressed as decimal 768. So, the high and low bytes of the starting address are:

```
HI=INT(768/256)=3
LO=768-HI*256=0
```

Next, POKE the address values in 1010 and 1011, and execute the CALL to update the pointers:

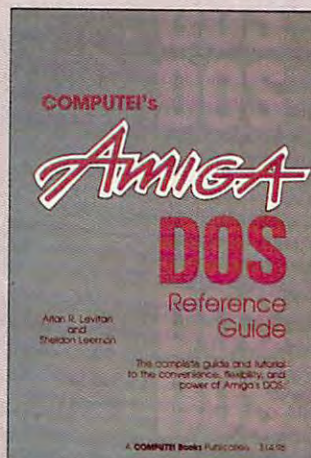
```
POKE 1010,LO
POKE 1011,HI
CALL -1169
```

When you hit RESET, the routine executes. Notice that the ML routine ends with JMP \$03D0. Because the routine is jumped to directly, it leaves no return address on the microprocessor's stack. If it had ended with an RTS, you'd wind up back in the machine language monitor after it's done. To avoid this unwanted result, you must exit with a JMP to the BASIC soft reentry point at \$03D0.

AMIGA

SUPPORT FROM COMPUTE! BOOKS

Everything for the Amiga. From BASIC beginner's guides to advanced programming handbooks, COMPUTE! offers you information-packed tutorials, reference guides, programming examples, ready-to-enter applications, and games to help you develop your computing skills on Commodore's Amiga.



COMPUTE!'s AmigaDOS Reference Guide

Arihan R. Levitan and Sheldon Leemon

A comprehensive tutorial and reference guide to the powerful AmigaDOS—the operating system underlying the Workbench and Intuition—this book offers information useful to every Amiga owner. It defines and illustrates all DOS commands, and shows you how to create file directories, access peripherals, run batch file programs, and avoid "disk shuffle." The screen- and line-oriented text editors are explained in detail. Numerous examples and techniques explain how to use AmigaDOS to make operating your Amiga both convenient and efficient.

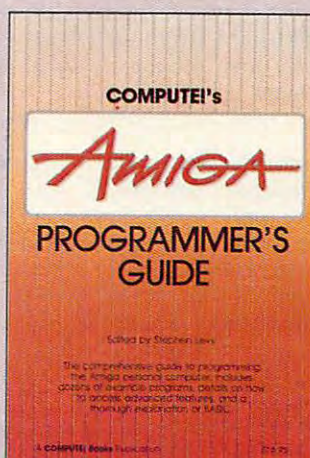
\$14.95 ISBN 0-87455-047-5

Elementary Amiga BASIC

C. Regena

Here's your introduction to the new and powerful BASIC on the Amiga personal computer. The Amiga's impressive graphics, animation, and sound can be unlocked with the right commands, and BASIC is the place to start. Complete descriptions of Amiga BASIC's commands, syntax, and organization take you from the beginner level to a full-fledged programmer. Plus, the book offers you ready-to-type-in programs and subroutines while showing you how to write your own programs. *There is a disk available which includes the programs in the book, \$12.95. This title is also available as a book/disk combination for \$29.95 (057-2).*

\$14.95 ISBN 0-87455-041-6



COMPUTE!'s Amiga Programmer's Guide

Edited

Your tutorial and reference manual to AmigaDOS, BASIC, Intuition, and other important software tools which accompany the new Amiga, *COMPUTE!'s Amiga Programmer's Guide* is a clear and thorough guide to the inner workings of this fascinating new-generation computer. The great speed of its 68000 microprocessor, coupled with the versatility of the Amiga-specific graphics and sound, makes the Amiga one of the most powerful computers available today. This book is the key to accessing the Amiga's speed and power.

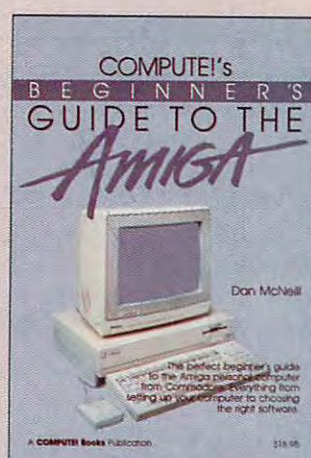
\$16.95 ISBN 0-87455-028-9

Advanced Amiga BASIC

Tom R. Halfhill and Charles Brannon

This guide to applications programming on Commodore's new Amiga contains everything an intermediate programmer requires to begin creating sophisticated software on this powerful machine, including several ready-to-type-in programs. Clear, yet comprehensive documentation and examples cover advanced BASIC commands, designing graphic applications, generating sound and music, using the Amiga's built-in speech synthesizer, creating a user interface, and programming the computer's peripherals. *There is a disk available which includes the programs in the book, \$15.95. (June release)*

\$16.95 ISBN 0-87455-045-9



COMPUTE!'s Beginners Guide to the Amiga

Dan McNeill

Written in a lively and entertaining style, this book teaches you everything a beginner needs to know to get started quickly with the Amiga from Commodore. You will learn about setting up the system, all the most popular types of software, and details about the hardware.

\$16.95 ISBN 0-87455-025-4

Inside Amiga Graphics

Sheldon Leemon

The Amiga, Commodore's powerful new computer, is an extraordinarily impressive graphics machine. Easy to use, the Amiga can produce color graphics and excellent animation. You'll find thorough descriptions of the computer's abilities and the hardware required to create a complete graphics system. Software, too, is central to the Amiga's power, and complete tutorials show you how to get the most from the machine. (June release)

\$16.95 ISBN 0-87455-040-8

COMPUTE!'s Kids and the Amiga

Edward H. Carlson

The latest in this bestselling series written by Edward Carlson, *COMPUTE!'s Kids and the Amiga*, will acquaint you with BASIC. Over 30 sections—all with instructor notes, lessons, assignments, and lively illustrations—entertain and amuse you as you learn to program your new computer. Clear writing and concise examples make it easy for anyone—children and adults alike—to painlessly learn BASIC. (May release)

\$14.95 ISBN 0-87455-048-3


Look for these books at your local book or computer store.
Or order directly from COMPUTE!.
Call toll-free 1-800-346-6767 (in NY 212-887-8525).

Please allow 4-6 weeks for delivery
after your order is received.

COMPUTE! Publications, Inc. 

Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
825 7th Avenue, 6th Floor, New York, NY 10019
Publishers of COMPUTE!, COMPUTE!'s Gazette, COMPUTE!'s Gazette Disk, COMPUTE!'s Books, and COMPUTE!'s Apple Applications

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England, and in Canada from McGraw-Hill, Ryerson Ltd., 330 Progress Ave., Scarborough, Ontario, Canada M1P 2Z5.

 www.commodore.ca

The same technique can be used to make the computer run a BASIC program when RESET is hit. Simply POKE the location of the Applesoft RUN routine into the RESET vector and call the ML from the first line of a BASIC program (for instance, with CALL -768). This could be done with the following program line:

```
10 POKE 1010,102: POKE 1011,213: CALL
-1169
```

In this case, you'd want to return to BASIC by ending the ML routine with RTS rather than JMP \$03D0.

Machine Language Division

I own a Commodore 64 and am teaching myself machine language. There is only one problem preventing me from completing my first useful program—I can't write a program to divide by ten. The objective is to take a number in the range 0-255 from memory and break it into each of its decimal parts. For example, 255 would break down into the digits 2, 5, and 5.

Kevin Owens

The 6502/6510 instruction set does not include a division instruction of any sort. Although it's possible to construct an ML routine for division, it is much easier and faster to use a method called successive subtraction. Here is the basic idea: Each power of ten from highest to lowest (in this case from 100 to 10 to 1) is subtracted from the number until you determine that the number has become negative. Each digit of the number is derived by counting the number of subtractions.

Below is the source code for a short program that displays any three-digit number at the upper-left corner of the screen. You'll need a machine language assembler to create the object code. (This program is written in PAL assembler format for the Commodore 64. Slight modifications are needed to assemble the code with a different assembler or to make the program work on a different 6502 computer.) Once the program is assembled, enter SYS 828: The program displays all the numbers from 0 to 255 in succession. The delay loop in line 130 gives you time to read each number. To see how fast numbers can be converted and displayed, remove this line and reassemble the program.

```
10 SYS 700: OPT 00: *828
20 TEMP = 2
30 LDA #0: STA 53281: LDA #147: J
SR $FFD2
40 LDA #1: STA 53281
50 LDA #0: STA TEMP
60 START INC TEMP: LDA TEMP
70 LDX #0
80 SUBAGAIN LDY #255
90 SUBMORE INY: SEC: SBC DIGITS,
X
100 BCS SUBMORE
```

```
110 ADC DIGITS, X
120 PHA: TYA: ORA #48: STA 1024, X
: PLA
130 INX: CPX #3: BNE SUBAGAIN
140 LDY #8: LDX #0: WT DEX: BNE W
T: DEY: BNE WT
150 JMP START
160 DIGITS .BYT 100, 10, 1
```

Atari BASIC Bugs

I have a very serious problem with my 16K Atari 600XL. Sometimes right after I've entered a line, the computer locks up and the only key that will work is SYSTEM RESET. But after I press RESET and enter another line, the computer locks up again.

Tak Lee

You're experiencing the latest incarnation of the infamous Atari BASIC lockup bug. This bug afflicts two versions of Atari BASIC: the original version, known as revision A, which was supplied as a cartridge for the 400, 800, and 1200XL; and revision B, which is built into the 600XL and 800XL computers. The lockup bug takes a slightly different form in these two versions of BASIC. In revision A, BASIC is unable to delete (move downward in memory) a block of memory whose size is an exact multiple of 256 bytes. Most users encounter the bug in the form of a keyboard lockup after deleting program lines, but it can affect the movement of strings as well. To illustrate, type in the following program:

```
A0 10 DIM A$(256), B$(256)
F0 20 FOR A=1 TO 256: A$(A,A)
="B": NEXT A
G0 30 B$=A$
I0 40 PRINT A$: PRINT B$
```

This creates a string variable, A\$, that consists of 256 B characters. Then it makes B\$ equal to A\$. When the program runs, you would expect it to print the letter B 512 times. Type RUN to see what happens instead. If you have revision A BASIC, the first 256 characters are correct, but the remaining characters are garbage. This occurs because BASIC's memory move routine was unable to move the value of A\$ into B\$ correctly. To confirm that the bug applies only to blocks of memory in multiples of 256, try changing the number 256 in lines 10 and 20 to some other value.

This bug was corrected when revision B BASIC was prepared for the 600XL and 800XL. However, Atari's programmers got carried away and applied the same correction to a routine which didn't need fixing—the routine which inserts (moves upward) blocks of memory, which happens when you add a BASIC program line. As a result, revision B BASIC has its own lockup bug which rears its ugly head when program lines are inserted instead of deleted. Ironically, the revision B bug

may occur even more often than the old one—you add program lines more often than you delete them. As an example, turn your 600XL or 800XL off and back on, then enter the following line in direct mode:

```
DIM A$(249): A$="TRASH"
```

Enter PRINT A\$ to see the variable value. Now enter this program line:

```
10 PRINT "THIS IS A TEST"
```

Before doing anything else, try to print the string again:

```
PRINT A$
```

If you have revision B, your computer should be locked up, and pressing RESET won't recover. For more details, see Bill Wilkinson's "Insight: Atari" columns in the May and June 1985 issues of COMPUTE!

The line-editing bug is not the only problem in early versions of Atari BASIC. Here's a list of some of the other bugs in revision A:

1. Because the value -0 is interpreted incorrectly, printing -0 yields garbage.

2. There's a problem with the precedence of the NOT operator which causes it to give unpredictable results in some cases. Type the following statement in direct mode when you have no other program in memory: PRINT NOT NOT 1.

3. LOCATE and GET statements may corrupt the internal buffer pointer. This can cause difficulties when trying to READ from DATA statements or when using the VAL function. If you have trouble with READ or VAL, use the statement X\$=STR\$(0) to reset the buffer pointer after a GET or LOCATE.

Revision B BASIC corrects some of the bugs from revision A, but also adds a few of its own. Here are some revision B bugs.

1. BASIC adds 16 bytes to the end of a program every time you LOAD or CLOAD it. After loading and saving the same program several times, you'll find that it has grown substantially. To remove the extra bytes, LIST the program to disk or tape and ENTER it back into memory. To avoid the problem, always use LIST/ENTER instead of SAVE/LOAD.

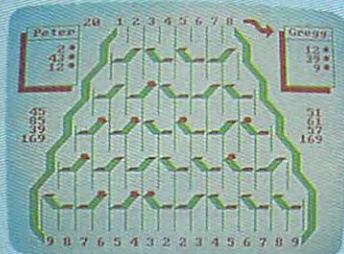
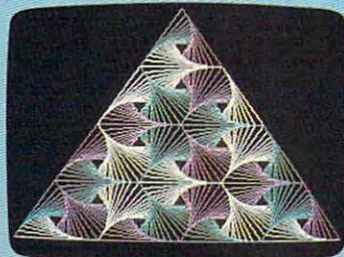
2. CLOAD and CSAVE commands fail to turn off the sound after they're done. Use END or the statement SOUND 0,0,0 to silence it.

3. Occasionally, an ERROR 9 (array or string DIM error) wrongly occurs in a program line that contains a DIM statement. You may be able to fix this condition by LISTing the program to disk and ENTERing it again.

One solution is to get the newest Atari BASIC cartridge, revision C, from Atari Corp., Customer Product Service, P.O. Box 61657, Sunnyvale, CA 94088. It costs \$15. This is the same BASIC built into the Atari 65XE and 130XE models.

©

GET UP TO 200 FUN-FILLED PROGRAMS EACH YEAR—when you subscribe now to COMPUTE!



PROOFREADER
Writing a machine language program that works on five different computers is no small task. The first hurdle is finding a safe place to put the code. Though the cassette buffer is an obvious choice, it's located in different places on various machines, and putting ML there creates problems for tape users. Instead, the Proofreader uses 256 bytes of BASIC programming space.

Before it installs the routine in memory, the Proofreader checks to see which computer you're using. Then it stores the ML at the bottom of BASIC memory and protects itself by moving the computer's start-of-BASIC pointer to a spot 256 bytes higher in memory. Once that's done, the Proofreader activates the ML routine and erases itself with NEH. Note that because the Proofreader overwrites its first few BASIC lines, it's critical not to delete anything from the first portion

Subscribe to COMPUTE! today through this special introductory money-saving offer, and you'll be getting a lot more than just another computer magazine. That's because each issue of COMPUTE! comes complete with up to 20 all-new, action-packed programs.

Subscribe now and you can depend on a steady supply of high quality, fun-filled programs like Hickory Dickory Dock, Switchbox, TurboDisk, Home Financial Calculator, Turbo Tape, SpeedScript, SpeedCalc, and hundreds of other educational, home finance, and game programs the entire family can use all year long.

The superb programs you'll find in each issue are worth much, much more than the low subscription price.

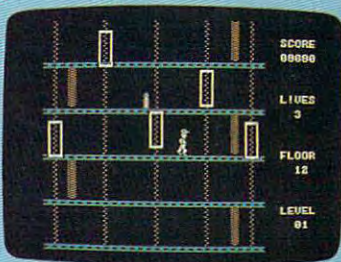
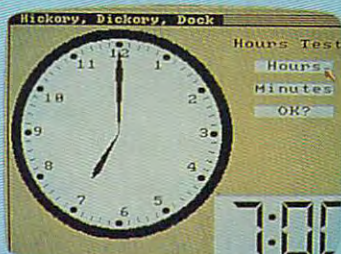
And there's more to COMPUTE! than just exciting new programs. Month after month, COMPUTE!'s superb articles deliver the latest inside word on everything from languages to interfaces...from programming to disk drives.

Whether you're a novice or an experienced user, COMPUTE! is the magazine for you. So subscribe today. Return the enclosed card or call 1-800-247-5470 (in Iowa, 1-800-532-1272).

Do it now.

SPREADSHEET: AB143+

| | 85 | 86 | 87 | 88 |
|----------------|----------|----------|----------|----|
| Month Expenses | | | | |
| January | 1817.22 | 582.23 | 429.39 | |
| February | 422.23 | 1842.23 | 628.88 | |
| March | 5821.88 | 4287.22 | 33.78 | |
| April | 1456.22 | 7728.23 | 6472.81 | |
| May | 512.22 | 2424.22 | 331.11 | |
| June | 1925.11 | 2384.22 | | |
| July | 242.45 | 2888.88 | 1656.55 | |
| August | 617.88 | 1727.88 | 1392.88 | |
| September | 1731.88 | 1758.88 | 742.88 | |
| October | 417.44 | 8888.88 | 2587.56 | |
| November | 677.33 | 7888.88 | 8122.11 | |
| December | 2578.88 | 2888.88 | 328.88 | |
| TOTALS | 25250.88 | 43448.35 | 26725.56 | |
| AVERAGE | 1427.14 | 1896.17 | 7469.85 | |



ACT NOW AND SAVE!

Getting Down to BASICS

In one form or another, the BASIC programming language has been around since before the dawn of personal computing. Easy to learn, simple to use, and useful for a wide variety of tasks, BASIC has opened the doors of computer programming for millions of people. Now, as a new generation of personal computers emerges, BASIC is continuing to evolve as a friendly and functional computer language—with a new look and feel.

Why BASIC? Why has this 22-year-old programming language—Beginner's All-purpose Symbolic Instruction Code—grown so immensely popular as an introduction to computer programming and as a general language? Despite its inherent limitations and numerous criticisms, BASIC remains the most widely taught and used language among computerists today.

The answers to these questions go back to a period before the advent of personal computing, in fact, years before a microcomputer was ever built. In the 1960s, the large mainframe computers ran programs by processing batches of punch cards—a system known as *batch processing*. These mainframes were machines that required a corps of trained operators to serve relatively untutored users. That average people would someday own and program powerful computers which fit on a desktop was unimaginable.

In those days, you didn't interact with a computer so much as present punch card offerings to it, and then wait for the computer to tell the computer operator to tell

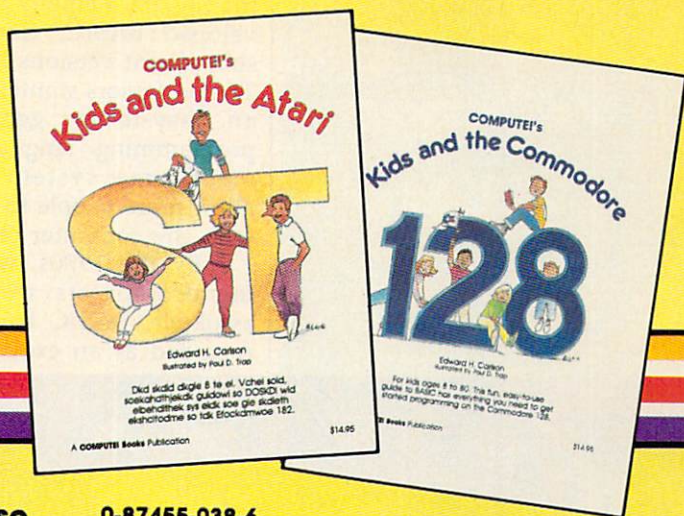
you the answer. More often than not, the first result was that a mistake somewhere in your batch of cards made an answer impossible until you corrected the error and then resubmitted the batch for another run. Not only was the process tedious, but it also meant waiting in line for access to the computer. A computer could serve only one user at a time.

Today, we tend to take for granted our ability to communicate quickly and easily with computers. The proliferation of personal computers has meant that individuals can have control over when and how often they work with a computer. And that kind of accessibility means that many people need a relatively easy way to communicate with computers without having to rely on other people as translators.

The original BASIC grew out of a project started back in 1964 by Dartmouth College mathematics professors Dr. John G. Kemeny and Dr. Thomas E. Kurtz. Kemeny and Kurtz were working with students on a timesharing project that would allow several people to gain simultaneous access to the university's mainframe computers. As a part of

NEW

COMPUTE! Books For Kids



Help your children learn the basics of computer programming with these two new entertaining and educational books from **COMPUTE!**

0-87455-038-6
\$14.95

0-87455-032-7
\$14.95

Each book contains easy-to-follow instructions, programming examples, quick reviews, and colorful illustrations. Written in COMPUTE!'s clear, easy-to-understand style, the books offer hours of entertainment while helping kids (and adults) learn to program in BASIC.

If you're acquainted with BASIC, you can easily write your own games and applications on Atari's ST or Commodore's 128 computers. Over 30 sections—all with instructor notes, lessons, assignments, and lively illustrations—entertain and amuse as you learn to use these powerful computers. *COMPUTE!'s Kids and the Atari ST* and *COMPUTE!'s Kids and the Commodore 128*, in the bestselling series from author Edward Carlson, are gentle introductions to programming your new computer. Clear writing and concise examples, both trademarks of this series, make it easy for anyone—child or adult—to learn BASIC painlessly.

Look for these and other books from **COMPUTE!** at your local book store or computer store. Or order directly from **COMPUTE!**

To order, call toll free in the US 1-800-346-6767 (In NY 212-265-8360) or mail the attached coupon with your payment to **COMPUTE! Books**, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

Please send me the following **COMPUTE!** books. My payment is enclosed.

_____ **COMPUTE!'s Kids and the Commodore 128**, (032-7) \$14.95 each

_____ **COMPUTE!'s Kids and the Atari ST**, (038-6) \$14.95 each

ALL ORDERS
MUST BE
PREPAID IN
U.S. FUNDS

Subtotal

NC residents add 4.5% sales tax

Shipping and handling per book
(In U.S. and surface mail, \$2.00 per
book; airmail, \$5.00 per book.)

Total amount enclosed

☐ Payment enclosed (check or money order)

☐ Charge ☐ MasterCard ☐ Visa ☐ American Express

Account No. _____ Exp. Date _____

(Required)

Name _____

Address _____

City _____

State _____ Zip _____

Please allow 4-6 weeks for delivery.

COMPUTE! Publications, Inc. 

One of the ABC Publishing Companies
825 7th Avenue, 6th Floor, New York, NY 10019

Publishers of *COMPUTE!*, *COMPUTE!'s Gazette*, *COMPUTE!'s Gazette Disk*, *COMPUTE! Books*, and *COMPUTE!'s Apple Applications*

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England and in Canada from Holt, Rinehart, & Winston, 55 Horner Avenue, Toronto, ON M8Z 4X6.

 www.commodore.ca

the project, Kemeny and Kurtz developed BASIC, from which all subsequent versions have evolved. The professors wanted BASIC to be an easy-to-use general-purpose programming language for their mainframe system that would allow more people to communicate with the computer on their own.

By the 1970s, when the first microcomputers were becoming available, BASIC had come to be regarded as an excellent language

gle standard. An involved program written in BASIC for one type of computer will rarely run on another type without at least some adjustments. Each version of BASIC embodies the strengths and weaknesses of the computer on which it runs, as well as the additions and deletions of those who adapted it from earlier versions. Some BASICs are so different from each other that they're almost like completely different languages.

Recently, this diversity led Kemeny and Kurtz to introduce what they call *True BASIC* (Addison-Wesley Publishing Company, Reading, Massachusetts) in late 1984. *True BASIC* is available for the IBM PC and compatibles, the Commodore Amiga, and the Apple Macintosh. In part, it's an attempt to deflect some of the criticism which has been aimed at BASIC over the years. Critics of BASIC often decry its lack of *structure*—it's not only possible, but quite easy, to write a BASIC program so disorganized that even the programmer cannot easily decipher it. On the other hand, BASIC's freedom from excessive structure-promoting rules is the very feature which attracts many programmers who prefer a more freeform style. Structured languages tend to encourage the production of more readable code, but also tend to impose more rules on the programmer. The debate over how rigidly structured a programming language should be is unlikely to end anytime soon.

True BASIC definitely leans toward the structured side. In fact, some of its new commands are almost identical to commands in Pascal, a popular structured language. Kemeny and Kurtz hope that *True BASIC*'s structure, speed, error-handling, mouse support, graphics, and easy transportability to other computers will establish it as a new standard.

As personal computers gained popularity, BASIC proved to be a fairly easy language to learn for most people. Since most versions of BASIC are *interpreters*, a programmer can enter a line of BASIC statements and test it immediately. Feedback is rapid because the computer interprets and carries out the commands instantly. But that also means that the computer has to



Developers of the original BASIC programming language, Dr. Thomas E. Kurtz (left) and Dr. John G. Kemeny, who now market a new version called *True BASIC*.

for personal programming and had already undergone several mutations. It became one of the first languages implemented on a personal computer when two college students—Paul Allen and Bill Gates—adapted it for the kit-built 4K RAM Altair in 1974. Allen and Gates later went on to found their own software company, Microsoft, Inc. Today, vastly improved descendants of the original Microsoft BASIC are available for almost all personal computers.

As BASIC became more widespread, it evolved in many different directions. Although the most popular version is Microsoft BASIC, none of the dozens of dialects adheres to a sin-

Commodore Software Sale

ORDER TODAY!

GAMES

| Access | | |
|----------------------------|---------|---------|
| 3500 MACH V (C) | \$34.95 | \$20.95 |
| 2128 MACH 128 (D) | 49.95 | 29.95 |
| 0451 BEACH HEAD (D) | 39.95 | 21.95 |
| 3038 BEACH HEAD II (D) | 49.95 | 23.95 |
| 0752 RAID OVER MOSCOW (D) | 39.95 | 26.95 |
| 0118 LEADER BOARD (D) | 39.95 | 24.95 |
| Accolade | | |
| 5950 HARBOLD (D) | \$29.95 | \$18.95 |
| 5952 LAW OF THE WEST (D) | 29.95 | 18.95 |
| 5954 FIGHT NIGHT (D) | 29.95 | 18.95 |
| 5956 PSI 5 TRADING CO. (D) | 29.95 | 18.95 |
| 5958 THE DAM BUSTERS (D) | 29.95 | 18.95 |

| Activision | | |
|------------------------------------|---------|---------|
| 0761 PITFALL II — LOST CAVERNS (D) | \$39.95 | \$20.95 |
| 0900 SPACE SHUTTLE (D) | 37.95 | 18.95 |
| 0932 ON FIELD FOOTBALL (D) | 39.95 | 20.95 |
| 0936 ON COURT TENNIS (D) | 39.95 | 20.95 |
| 0940 GHOSTBUSTERS (D) | 39.95 | 23.95 |
| 3580 GREAT AMERICAN R. RACE (D) | 29.95 | 18.95 |
| 3582 MASTER OF THE LAMPS (D) | 29.95 | 20.95 |
| 3584 COUNTDOWN/SHUTDOWN (D) | 29.95 | 20.95 |
| 3588 MINDSHADOW (D) | 29.95 | 18.95 |
| 3590 STAR LEAGUE BASEBALL (D) | 29.95 | 20.95 |
| 3592 ALCAZAR (D) | 29.95 | 20.95 |
| 5196 LITTLE PEOPLE PROJECT (D) | 34.95 | 22.95 |
| 5198 FAST TRACKS (D) | 34.95 | 20.95 |
| 5202 GAMESMAKER (D) | 39.95 | 24.95 |
| 3585 COMPLETE FIREWORKS KIT (D) | 34.95 | 22.95 |
| 3612 ALTER EGO (D) | 49.95 | 29.95 |
| 3614 BORROWED TIME (D) | 29.95 | 18.95 |
| 5200 HACKER (D) | 29.95 | 18.95 |
| 1572 STAR RANK BOXING (D) | 29.95 | 20.95 |

| Avalon Hill | | |
|--------------------------------|---------|---------|
| 0396 SUPER BOWL SUNDAY (D) | \$35.00 | \$22.95 |
| 3572 SPITFIRE 40 (D) | 35.00 | 22.95 |
| 5138 STATUS PRO BASEBALL (D) | 35.00 | 22.95 |
| 5250 MISSION / THUNDERHEAD (D) | 25.00 | 17.95 |
| 5146 JUPITER MISSION (D) | 35.00 | 22.95 |
| 5252 GULF STRIKE (D) | 30.00 | 19.95 |
| 5254 MACBETH (D) | 25.00 | 17.95 |
| 2375 COMPUTER TITLE BOUT (D) | 30.00 | 19.95 |
| 0860 TOURNAMENT GOLF (D) | 29.95 | 18.95 |
| 5140 BLACK THUNDER (D) | 19.95 | 14.95 |

| Broderbund | | |
|--------------------------------|---------|---------|
| 2903 LODE RUNNER (D) | \$34.95 | \$19.95 |
| 2905 KARATEKA (D) | 29.95 | 23.95 |
| 3038 CHAMPION LODE RUNNER (D) | 34.95 | 26.95 |
| 5158 BANK STREET WRITER (D) | 49.95 | 32.95 |
| 5330 BANK STREET SPELLER (D) | 49.95 | 32.95 |
| 5332 BANK STREET FILER (D) | 49.95 | 32.95 |
| 5334 BANK STREET MAILER (D) | 49.95 | 32.95 |
| 2540 PRINT SHOP (D) | 44.95 | 27.95 |
| 2542 GRAPHIC LIBRARY NO. 1 (D) | 24.95 | 15.95 |
| 3898 GRAPHIC LIBRARY NO. 2 (D) | 24.95 | 15.95 |
| 3897 GRAPHIC LIBRARY NO. 3 (D) | 24.95 | 15.95 |
| 2910 PRINT SHOP COMPANION (D) | 39.95 | 24.95 |
| 5160 MUSIC SHOP (D) | 44.95 | 28.95 |
| 5170 LODE RUNNERS RESCUE (D) | 29.95 | 20.95 |

| Electronic Arts | | |
|-----------------------------------|---------|---------|
| 3830 DR. J & LARRY BIRD (D) | \$29.95 | \$23.95 |
| 3832 FINANCIAL COOKBOOK (D) | 39.95 | 27.95 |
| 3834 MAIL ORDER MONSTERS (D) | 34.95 | 22.95 |
| 3840 THE SEVEN CITIES OF GOLD (D) | 29.95 | 23.95 |
| 3842 SKY FOX (D) | 29.95 | 23.95 |
| 5176 CARRIERS AT WAR (D) | 42.95 | 32.95 |
| 5178 REACH FOR THE STARS II (D) | 37.95 | 28.95 |
| 5180 HEART OF AFRICA (D) | 29.95 | 23.95 |
| 5182 MOVIE MAKER (D) | 29.95 | 23.95 |
| 5184 EUROPE ABLAZE (D) | 42.95 | 34.95 |
| 5186 M.U.L.E. (D) | 19.95 | 16.95 |
| 5188 MURDER ON ZINDERNEUF (D) | 19.95 | 16.95 |
| 5190 MUSIC CONSTRUCTION SET (D) | 19.95 | 16.95 |
| 5192 PINBALL CONSTRUCTION SET (D) | 19.95 | 16.95 |
| 5194 RACING CONSTRUCTION SET (D) | 29.95 | 22.95 |
| 3601 SUPER BOULDERDASH (D) | 29.95 | 22.95 |
| 3600 TOUCHDOWN FOOTBALL (D) | 29.95 | 22.95 |



Phone Orders

(T) Tape, (C) Cartridge, (D) Disk.

Datasoft

| | | |
|------------------------|---------|---------|
| 3025 BRUCE LEE (D) | \$34.95 | \$19.95 |
| 3026 PAC-MAN (D) | 34.95 | 17.95 |
| 3027 MIGHTY CONAN (D) | 34.95 | 22.95 |
| 3028 MR. DO! (D) | 34.95 | 18.95 |
| 3029 DIG DUG (D) | 34.95 | 18.95 |
| 3032 POLE POSITION (D) | 34.95 | 18.95 |
| 5218 THE GOONIES (D) | 29.95 | 18.95 |
| 5220 ZORRO (D) | 29.95 | 18.95 |

Epyx

| | | |
|----------------------------------|---------|---------|
| 0337 WORLD'S GREAT FOOTBALL (D) | \$39.95 | \$23.95 |
| 0338 WINTER GAMES (D) | 39.95 | 20.95 |
| 0339 THE IDOLON (D) | 39.95 | 20.95 |
| 0340 KORONIS RIFT (D) | 39.95 | 20.95 |
| 0360 JET COMBAT SIMULATION (D) | 39.95 | 20.95 |
| 0364 SUMMER OLYMPIC GAMES (D) | 39.95 | 18.95 |
| 0365 WORLD'S GREAT BASEBALL (D) | 34.95 | 22.95 |
| 0382 SUMMER OLYMPIC GAMES II (D) | 39.95 | 20.95 |
| 0750 PITSTOP II (D) | 39.95 | 22.95 |
| 2046 IMPOSSIBLE MISSION (D) | 34.95 | 16.95 |
| 2066 ROBOTS OF DAWN (D) | 39.95 | 15.95 |
| 2070 BARBIE (D) | 39.95 | 18.95 |
| 2074 G.I. JOE (D) | 39.95 | 18.95 |
| 3005 BALLBLAZER (D) | 29.95 | 20.95 |
| 3006 RESCUE ON FRACALUSI (D) | 29.95 | 20.95 |
| 1556 MOVIE MONSTER GAME (D) | 39.95 | 24.95 |
| 1557 MICROSOFT MULTIPLAN (D) | 59.95 | 39.95 |
| 1558 PROG. BASIC TOOLKIT (D) | 44.95 | 29.95 |
| 1559 VORPAL UTILITY KIT (D) | 34.95 | 22.95 |

Strategic Simulations, Inc.

| | | |
|---------------------------------|---------|---------|
| 2995 RDF 1985 (D) | \$34.95 | \$20.95 |
| 2997 GEOPOLITIQUE (D) | 39.95 | 23.95 |
| 3008 RINGSIDE SEAT (D) | 39.95 | 23.95 |
| 3010 IMPERIUM GALACTIC (D) | 39.95 | 23.95 |
| 3011 CARTELS AND CUTTHROATS (D) | 39.95 | 23.95 |
| 3012 RAILS WEST (D) | 39.95 | 23.95 |
| 3014 PROFESSIONAL TOUR GOLF (D) | 39.95 | 23.95 |
| 3015 50 MISSION CRUSH (D) | 39.95 | 23.95 |
| 3016 PRESIDENT ELECT (D) | 39.95 | 23.95 |
| 3017 BROADSIDES (D) | 39.95 | 24.95 |
| 3018 COMPUTER QUARTERBACK (D) | 39.95 | 24.95 |
| 3020 COMPUTER AMBUSH (D) | 59.95 | 37.95 |
| 3021 COMPUTER BASEBALL (D) | 39.95 | 23.95 |
| 3031 FIELD OF FIRE (D) | 39.95 | 23.95 |
| 5154 KAMPFGROUPE (D) | 59.95 | 34.95 |
| 5156 COLONIAL CONQUEST (D) | 39.95 | 23.95 |
| 3768 U.S.A.A.F. (D) | 59.95 | 36.95 |
| 1560 SIX GUN SHOOTOUT (D) | 39.95 | 23.95 |
| 1561 BATTLE OF ANTIETAM (D) | 49.95 | 31.95 |
| 1562 BATTALION COMMANDER (D) | 39.95 | 23.95 |
| 1563 PANZER GRENADIER (D) | 39.95 | 23.95 |
| 1564 NORWAY 1985 (D) | 34.95 | 20.95 |
| 1565 MECH BRIGADE (D) | 59.95 | 36.95 |
| 1567 BATTLEGROUP (D) | 59.95 | 37.95 |

BUSINESS

Softsync

| | | |
|--------------------------------|---------|---------|
| 5930 ACCOUNTANT, INC. (D) C128 | \$99.95 | \$64.95 |
| 5932 PERSONAL ACCOUNTANT (D) | 34.95 | 26.95 |
| 5934 MODEL DIET (D) | 29.95 | 23.95 |
| 5936 TRIO (D) C128 | 49.95 | 45.95 |
| 5938 KID PRO QUO (D) | 29.95 | 23.95 |
| 5940 DESK MANAGER (D) C128 | 39.95 | 28.95 |

Timeworks

| | | |
|--|---------|---------|
| 0176 INVENTORY MANAGE (D) | \$69.95 | \$38.95 |
| 0180 ACCOUNTS RECEIVABLE/INVOICING (D) | 69.00 | 38.95 |
| 0182 ACCOUNTS PAYABLE/CHECKWRITING (D) | 69.00 | 38.95 |
| 0184 PAYROLL MANAGEMENT (D) | 69.00 | 38.95 |
| 0188 GENERAL LEDGER (D) | 69.00 | 38.95 |
| 0928 EVELYN WOOD SPEED READ (D) | 69.95 | 32.95 |
| 5022 WORDWRITER & DATA MANAGER II (D) | 98.00 | 49.00 |
| 5026 SWIFTCALC/SIDEWAYS (D) | 49.95 | 32.95 |

Business Continued

C128 Software From Timeworks

| | | |
|-------------------------------------|---------|---------|
| 5022 WORD WRITER/ SPELL CHECKER (D) | \$69.95 | \$59.95 |
| 5024 DATA MANAGER II (D) | 69.95 | 49.95 |
| 5026 SWIFTCALC WITH SIDEWAYS (D) | 69.95 | 49.95 |
| 5030 PARTNER (D) | 59.95 | 39.95 |
| 3048 SYLVIA PORTER (D) | 69.95 | 39.95 |

EDUCATION

American Educational Computer

| | | |
|--------------------------------|---------|---------|
| 2482 ELEM. SCIENCE FACTS (D) | \$29.95 | \$14.95 |
| 2492 VOCABULARY WORD BUILD (D) | 29.95 | 14.95 |
| 2493 GRAMMAR WORD SKILLS (D) | 29.95 | 14.95 |
| 2494 WORLD GEOGRAPHY FACTS (D) | 29.95 | 14.95 |
| 2495 SPANISH VOCAB. SKILLS (D) | 29.95 | 14.95 |
| 2496 FRENCH VOCAB. SKILLS (D) | 29.95 | 14.95 |
| 2497 WORLD HISTORY (D) | 29.95 | 14.95 |
| 2498 U.S. HISTORY FACTS (D) | 29.95 | 14.95 |
| 2499 BIOLOGY FACTS (D) | 29.95 | 14.95 |
| 2519 U.S. GEOGRAPHY FACTS (D) | 29.95 | 14.95 |
| 2520 U.S. GOVERNMENT FACTS (D) | 29.95 | 14.95 |
| 2521 AEC SPELLING (D) | 39.95 | 24.95 |
| 3745 PHONICS (D) | 39.95 | 24.95 |
| 3747 LEARN TO READ (D) | 39.95 | 24.95 |
| 3749 READING COMPREHENSION (D) | 39.95 | 24.95 |

Designware

| | | |
|---------------------------------------|---------|---------|
| 0824 GRAMMAR EXAMINER (D) | \$39.95 | \$24.95 |
| 0828 SPELLKAZAM (D) | 34.95 | 9.95 |
| 0832 STATES & TRAITS (D) | 44.95 | 27.95 |
| 0836 SPELLICOPTER (D) | 39.95 | 22.95 |
| 0840 CREATURE CREATOR (D) | 34.95 | 9.95 |
| 0844 TRAP-A-ZOID (D) | 39.95 | 9.95 |
| 2518 THE BODY TRANSPARENT (D) | 44.95 | 27.95 |
| 2517 EUROPEAN NATIONS & LOCATIONS (D) | 44.95 | 19.95 |
| 2062 MATH MAZE (D) | 39.95 | 22.95 |
| 5100 ALGEBRA 1 (D) | 39.95 | 19.95 |
| 5102 REMEMBER (D) | 69.95 | 49.95 |
| 5104 WEBSTER'S NUMBERS (D) | 39.95 | 19.95 |
| 5105 SPELLING & READ PRIMER (D) | 39.95 | 19.95 |
| 5106 ALGEBRA 2 (D) | 39.95 | 19.95 |
| 5107 ALGEBRA 3 (D) | 39.95 | 19.95 |

Mindscape

| | | |
|--|-------|-------|
| 5108 KEYBOARD CADET (D) | 39.95 | 25.95 |
| 5110 BANK STREET MUSIC WRITER (D) | 39.95 | 25.95 |
| 5112 CROSSWORD MAGIC (D) | 49.95 | 29.95 |
| 5114 THE PERFECT SCORE (D) | 69.95 | 45.95 |
| 5116 COLORED/RAINBOW BRIDE (D) | 34.95 | 18.95 |
| 5118 THE HALLEY PROJECT (D) | 39.95 | 22.95 |
| 5120 INDIANA JONES IN THE LOST KINGDOM (D) | 29.95 | 18.95 |
| 5122 BANK STREET STORYBOOK (D) | 39.95 | 22.95 |
| 5910 THE DOLPHIN'S RUNE (D) | 29.95 | 18.95 |
| 5912 THE LUSCHER PROFILE (D) | 39.95 | 22.95 |
| 5914 QUAKE MINUS ONE (D) | 29.95 | 16.95 |
| 5916 THE LORDS OF MIDNIGHT (D) | 29.95 | 16.95 |
| 5918 SHADOWFIRE (D) | 29.95 | 16.95 |
| 3702 BOP 'N' WRESTLE (D) | 29.95 | 21.95 |
| 3690 INFILTRATOR (D) | 29.95 | 21.95 |

Weekly Reader Buy 1 Get One Free!

| | | |
|----------------------------------|-------|-------|
| 2512 STICKYBEAR NUMBERS (D) | 34.95 | 14.95 |
| 2513 STICKYBEAR BASKETBOUNCE (D) | 34.95 | 14.95 |
| 2514 STICKYBEAR OPPOSITES (D) | 34.95 | 14.95 |
| 2515 STICKYBEAR ABC (D) | 34.95 | 14.95 |
| 2516 STICKYBEAR SHAPES (D) | 34.95 | 14.95 |
| 2600 PIC BUILDER (D) | 29.95 | 14.95 |
| 5126 STICKYBEAR SPELLGRABBER (D) | 29.95 | 14.95 |
| 5128 STICKYBEAR TOWN BUILDER (D) | 29.95 | 14.95 |
| 5130 STICKYBEAR MATH (D) | 29.95 | 14.95 |
| 5132 STICKYBEAR READING (D) | 29.95 | 14.95 |
| 5129 STICKYBEAR TYPING (D) | 29.95 | 14.95 |

Add \$3.00 for shipping, handling and insurance. Illinois residents please add 6 1/2% tax. Add \$6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES, EXCEPT CANADA. Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Prices & availability subject to change without notice. VISA — MASTER CARD — C.O.D. No. C.O.D. to Canada, APO-FPO

PROTECTO

We Love Our Customers

22292 N. Pepper Rd., Barrington, Illinois 60010

312/382-5244 to order

www.commodore.ca

interpret every line of code while the program is running.

An alternative approach is a *compiler*. Popular compiled languages include Pascal, FORTRAN, C, COBOL, and some BASICs. Running a program with a compiler requires two steps. First, the compiler interprets all the commands in the program without carrying them out and creates a new version of the program on disk called *object code*, *p-code*, or *run-time code*. This file, incomprehensible to human eyes, is quite similar to a program written

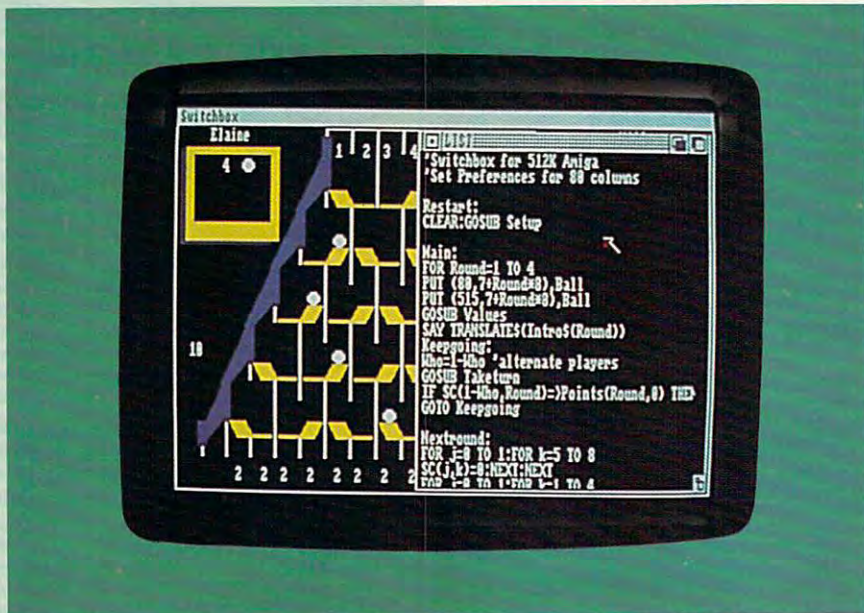
becoming popular on personal computers, since until recently most machines were limited to 64K of RAM. Because BASIC interpreters can be squeezed into as little as 8K of RAM, BASIC was the logical choice for the first generation of microcomputers.

Most personal computer owners who are interested in programming quickly grow accustomed to the version of BASIC that comes with their machines. Some BASICs are built into the computer's Read Only Memory (ROM), while others are supplied on plug-in ROM cartridges or floppy disks. But there have been significant variations of BASIC even for the same brands of computers. Besides that, additional versions of BASIC are often made available by independent sources, as are packages which add enhancements to existing BASICs.

For example, Applesoft BASIC is a version of Microsoft BASIC that's used by Apple II-series computers. Integer BASIC, an earlier BASIC from Apple, was available in ROM on the original Apple II. Although faster than Applesoft BASIC, Integer BASIC doesn't allow floating-point math operations (the use of fractions) as does Applesoft. The Apple II+ machine came with Applesoft BASIC in ROM, while the Apple IIe and IIc computers have Applesoft BASIC on built-in language cards. (The II+ can add a language card, too.)

Atari computers have had several different versions of BASIC available, as well as optional third-party BASIC languages. The Atari 400, 800, and 1200XL computers come with an 8K BASIC ROM cartridge, while the 600XL, 800XL, and 130XE computers have later revisions of BASIC built into ROM. There are also alternatives to Atari BASIC, such as Microsoft BASIC from Atari and BASIC XL and BASIC A+ from Optimized Systems Software (OSS) in San Jose, California.

Commodore 64 owners are familiar with the BASIC 2.0 version in their computers, the same version that appeared in the earlier VIC-20. Prior to 2.0, the earlier Commodore PET computer included a version 4.0. A variation



In Amiga BASIC line numbers are unnecessary, and are replaced with labels that indicate subroutines and program divisions.

in machine language—it's a complex pattern of bits which is the only language that any computer really understands. After the compilation is completed, the object code can be run. Since all the commands in the program have already been interpreted by the compiler, the object code runs much faster than a program which must be interpreted one command at a time.

Unfortunately, the two-step process of compiling can take many minutes, which is frustrating for programmers—especially beginners. If the program contains an error, the whole process has to be repeated. It's not as frustrating as the old batch processing, but it's a step back in that direction.

Compiled languages also require more computer power than interpreted languages. They need faster processors and more memory—sometimes 512K of Random Access Memory (RAM) is scarcely enough for a compiler. This has largely prevented compilers from

C-128™ AND ATARI ST™ REQUIRED READING



C-128 INTERNALS
Detailed guide presents the 128's operating system, explains, graphic chips, Memory Management Unit, 80-col graphics, commented ROM listings. A book no 128 programmer should be without. 500pp \$19.95

C-128 TRICKS & TIPS
Practical, easy-to-use techniques for C-128 users. 80 column graphics, windowing, memory layout, autostarting, Kernel routines, custom character sets, software protection, sprites, command extensions. 300pp. \$19.95

1571 INTERNALS
Essential reference for 1571 users. Sequential & relative files, direct access commands, "foreign" disk formats, CP/M and the 1571, zero page and error messages. Fully commented DOS ROM listings. 500pp. \$19.95

128 Basic Training Guide
Thorough introduction to programming for the beginner. Topics include: problem analysis, complete description of BASIC commands with examples and problems, structured programming, monitor commands. 200pp \$16.95

C-128 PEEKS & POKES
Presents dozens of programming quick-hitters. Easy and useful techniques on the operating system, zero-page, pointers & stack, 640x200 graphics, Z80 and 8502 machine language, BASIC interpreter. 200pp \$19.95

CP/M ON THE C-128
Finally! CP/M revealed on the C-128. Simple explanations of the operating system, resident/transient commands, memory usage, system disk, using PIP, CP/M utility programs, commented Z80 ROM listing & more. \$19.95



ST INTERNALS
Essential guide to learning the inside information of the ST. Detailed descriptions of sound and graphics chips, internal hardware, I/O ports, using GEM. Commented BIOS listing. An indispensable reference for your ST library. 450pp. \$19.95

GEM Programmer's Ref.
For serious programmers in need of detailed information on GEM. Written with an easy to understand format. All GEM examples are written in C and assembly language. Required library addition that no serious programmer should be without. 450pp. \$19.95

ST TRICKS & TIPS
Fantastic collection of programs and info for the ST. Complete programs include: super-fast RAM disk; time-saving printer spooler; color print hardcopy; plotter output hardcopy; creating accessories. Money saving tricks and tips. 260 pp. \$19.95

ST GRAPHICS & SOUND
Detailed guide to understanding graphics & sound on the ST. 2D & 3D function plotters, Moiré patterns, various resolutions and graphic memory, fractals, waveform generation. Examples written in C, LOGO, BASIC and Modula2. \$19.95

ST LOGO GUIDE
Take control of your ST by learning ST LOGO—the easy to use, powerful language. Topics include: file handling, recursion-Hilbert & Sierpinski curves, 2D and 3D function plots, data structure, error handling. Helpful guide for ST LOGO users. \$19.95

ST PEEKS & POKES
Enhance your programs with the examples found within this book. Explores using different languages BASIC, C, LOGO and machine language, using various interfaces, memory usage, reading and saving from and to disk, more. \$16.95

Commodore 128 is a trademark of Commodore Business Machines Inc.
The ATARI logo and ATARI ST are trademarks of Atari Corp.

Abacus Software

P.O. Box 7219 Dept. C7 Grand Rapids, MI 49510 - Telex 709-101 - Phone (616) 241-5510

Optional diskettes available for all book titles - \$14.95 each. Other books & software also available. Call for the name of your nearest dealer. Or order directly from ABACUS using your MC, Visa or Amex card. Add \$4.00 per order for shipping. Foreign orders add \$10.00 per book. Call now or write for your free catalog. Dealer inquiries welcome—over 1400 dealers nationwide.

 www.commodore.ca

between BASIC 2.0 and BASIC 4.0, called BASIC 3.5, was included in the Plus/4 and 16 computers. And the 128 includes a powerful version of BASIC, 7.0, that contains virtually all of the commands of the earlier BASICs.

But despite the differences among these forms of BASIC, they're all fairly similar in their organization. They're interpreted, giving immediate feedback to the user; they all use similar *commands*, *variables*, and *functions*; each line of BASIC begins with a line number;

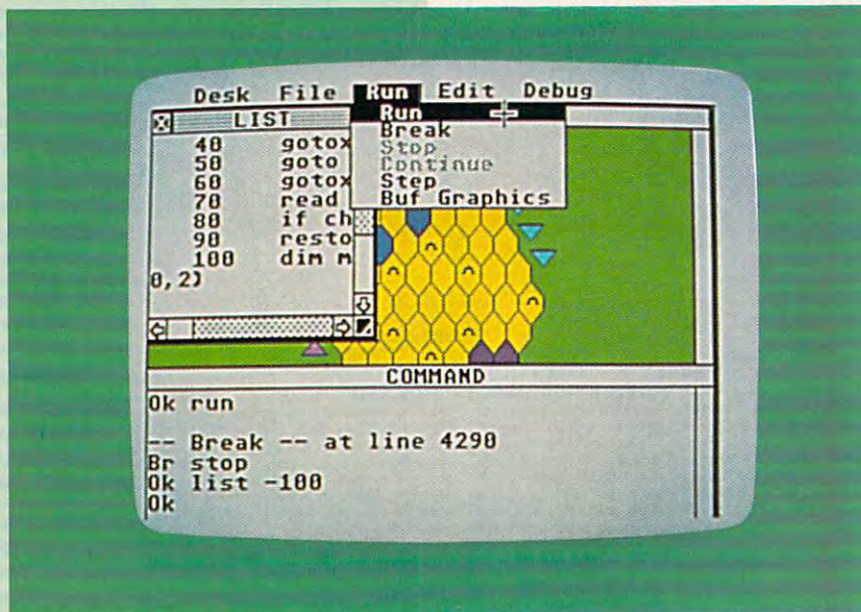
tosh BASIC abandon the line numbers used in previous BASICs. Instead, meaningful labels are used that identify sections of code and subroutines. Although ST BASIC does have line numbers, you can put labels within lines and direct subroutines to those labels.

The programming environment changes as well. Windows—with separate areas for your commands, the program listing, and the program output—take the place of the single screen you may be used to. With this system, you can actually see the program run while the program's code stays visible. Using a mouse, you can click on menu items like RUN and LIST instead of typing them in.

Macintosh BASIC, for example, offers several windows: a Command window to take your directions; two List windows, allowing you to have two different parts of the program onscreen at the same time; and an Output window which allows you to see the results of your programming. There are also programming tools that simplify your efforts. TRACE MODE, a debugging tool that can be switched on or off, highlights whatever line in your program is currently executing. The new BASICs also generally support the currently popular mouse environment, allowing you to create your own custom-designed windows, pull-down menus, and dialog boxes.

ST BASIC is fundamentally similar to the BASICs you may have used on your eight-bit computer, but offers accessibility to windows, drop-down menus, and graphic icons from the GEM Desktop environment much like Amiga BASIC and Macintosh BASIC. There are actually four windows on the ST BASIC screen: Output, List, Command, and, hidden behind the first three, an Edit window.

It would, of course, be precipitate to conclude that the new BASICs represent the ultimate in man-machine communication. Rather, they seek to offer a higher level of power, ease, and efficiency to the computer programmer. Computer languages are continually evolving as the search continues for ever more effective methods by which man can interact with his increasingly intelligent inventions. ©



ST BASIC has line numbers, but the GEM operating environment includes multiple windows, drop-down menus, and icons much like AmigaBASIC and Macintosh BASIC.

BASIC doesn't usually permit your computer to crash, so it's friendly to programmers; and access to printers and other peripherals is relatively easy to accomplish.

Within the past year, as the new Commodore Amiga and Atari ST computers have joined the Apple Macintosh—machines based on the more powerful 68000 microprocessor—computer users have been confronted with new BASIC languages that have several important differences from earlier versions. The Macintosh and the Amiga have BASIC languages that are almost identical, both created by Microsoft. The Amiga was initially released with a BASIC language called ABASIC, but that was superseded by the Microsoft version, called Amiga BASIC. The Atari ST, at this writing, has an ST BASIC from Atari, as well as several other versions of BASIC that should be available from third-party companies by the time you read this.

Both Amiga BASIC and Macin-

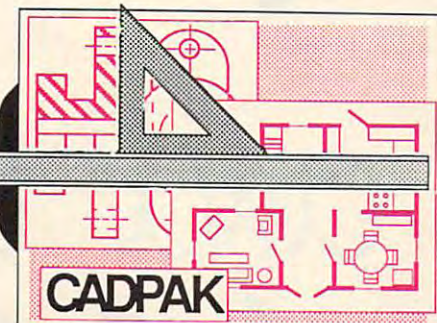
'128TM and C-64TM

SENSATIONAL SOFTWARE

FAST COMPILER

Make your BASIC programs run LIGHTNING SPEED!

The complete compiler and development package. Speed up your programs 5x to 35x. Many options: flexible memory management; choice of compiling to machine code, compact p-code or both. '128 version: 40 or 80 column monitor output and FAST-mode operation. '128 Compiler's extensive 80-page programmer's guide covers compiler directives and options, two levels of optimization, memory usage, I/O handling, 80 column hi-res graphics, faster, higher precision math functions, speed and space saving tips, more. A great package that no software library should be without. **128 Compiler \$59.95**
64 Compiler \$39.95



Remarkably easy-to-use interactive drawing package for accurate graphic designs. New dimensioning features to create exact scaled output to all major dot-matrix printers. Enhanced version allows you to input via keyboard or high quality lightpen. Two graphic screens for COPYING from one to the other. DRAW, LINE, BOX, CIRCLE, ARC, ELLIPSE available. FILL objects with preselected PAT-TERNS; add TEXT; SAVE and RECALL designs to/from disk. Define your own library of symbols/objects with the easy-to-use OBJECT MANAGEMENT SYSTEM—store up to 104 separate objects. **C-128 \$59.95**
C-64 \$39.95

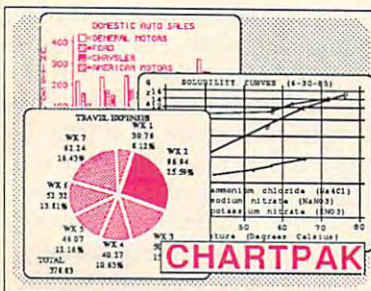


Fast loading (8 sec. 1571, 18 sec. 1541); Two standard I/O libraries—math functions (sin, cos, sqrt, etc.) & 20+ graphic commands (line, fill, dot, etc.). **C-128 \$79.95**
C-64 \$59.95

For school or software development. Learn C on your Commodore with our in-depth tutorial. Compile C programs into fast machine language. C-128 version has added features: Unix™-like operating system; 60K RAM disk for fast editing and compiling Linker combines up to 10 modules; Combine M/L and C using CALL; 51K available for object code;



Not just a compiler, but a complete system for developing applications in Pascal with graphics and sound features. Extensive editor with search, replace, auto, renumber, etc. Standard J & W compiler that generates fast machine code. If you want to learn Pascal or to develop software using the best tools available—SUPER Pascal is your first choice. **C-128 \$59.95**
C-64 \$59.95



Easily create professional high quality charts and graphs without programming. You can immediately change the scaling, labeling, axis, bar filling, etc. to suit your needs. Accepts data from CalcResult and MultiPlan. C-128 version has 3X the resolution of the '64 version. Outputs to most printers. **C-128 \$39.95**
C-64 \$39.95

PowerPlan

One of the most powerful spreadsheets with integrated graphics. Includes menu or keyword selections, online help screens, field protection, windowing, trig functions and more. PowerGraph, the graphics package, is included to create integrated graphs and charts. **C-64 \$39.95**

Technical Analysis System for the C-64 **\$59.95**
Ada Compiler for the C-64 **\$39.95**
VideoBasic Language for the C-64 **\$39.95**

OTHER TITLES AVAILABLE:

COBOL Compiler

Now you can learn COBOL, the most widely used commercial programming language, and learn COBOL on your 64. COBOL is easy to learn because its easy to read. COBOL Compiler package comes complete with Editor, Compiler, Interpreter and Symbolic Debugger. **C-64 \$39.95**

Personal Portfolio Manager

Complete portfolio management system for the individual or professional investor. Easily manage your portfolios, obtain up-to-the-minute quotes and news, and perform selected analysis. Enter quotes manually or automatically through Warner Computer Systems. **C-64 \$39.95**

Xper

XPER is the first "expert system" for the C-128 and C-64. While ordinary data base systems are good for reproducing facts, XPER can derive knowledge from a mountain of facts and help you make expert decisions. Large capacity. Complete with editing and reporting. **C-64 \$59.95**

C-128 and C-64 are trademarks of Commodore Business Machines Inc.
Unix is a trademark of Bell Laboratories

Abacus Software

P.O. Box 7219 Dept. C6 Grand Rapids, MI 49510 - Telex 709-101 - Phone (616) 241-5510

Call now for the name of your nearest dealer. Or to order directly by credit card, MC, AMEX or VISA call (616) 241-5510. Other software and books are available—Call and ask for your free catalog. Add \$4.00 for shipping per order. Foreign orders add \$12.00 per item. Dealer inquiries welcome—1400+ nationwide.

www.commodore.ca

C and the 68000

Kathy Yakal, Assistant Features Editor

A programming language written almost 15 years ago for a specific purpose—to develop an operating system for a mainframe computer—has been getting a lot of attention in the microcomputer community lately. C, a language described by many as simple, elegant, and powerful, is especially well-suited for programming the 68000 chip housed in the Commodore Amiga, Atari ST, and Apple Macintosh.

C is a beguiling language. In structure and difficulty, it falls somewhere between machine language and a higher level language such as BASIC or Pascal. C was developed in 1972 by Dennis Ritchie, who wrote it specifically to design the UNIX operating system running on the PDP-11 (a mainframe computer). Though some mainframe and minicomputer programmers have chosen to use it in the years since, it's enjoying a renewed popularity with the advent of the new 68000-based machines: the Atari ST, the Apple Macintosh, and the Commodore Amiga.

C is the language of choice for the Amiga and the ST, say many

programmers, for four main reasons. First, its basic command structure is quite concise, but can be extended by individual programmers for specific functions. Second, its relative closeness to actual machine language gives the programmer tremendous power. Third, there is a harmony between the 68000 microprocessor and the C language; 68000 machine language itself supports C constructs, thanks in part to the similarities between the PDP-11 and the 68000. Finally, and perhaps most important for the home computer market, programs written in C on one 68000-based machine can be more easily and quickly transported to another computer than can programs written in many other languages.

Programmers consider several factors when deciding what language to use. Of course, everyone has favorites based on personal experience, but the physical capabilities of individual computers and the type of application being written necessarily create some restrictions.

Just as some computers are designed to support specific lan-

guages, some languages have been developed to support specific applications. You *could* accomplish almost anything in almost any language, but there is significant variability between languages in the efficiency of program writing and executing. Some languages are simply more appropriate than others for specific jobs.

One of the first high-level languages, FORTRAN, was written in 1954, and is geared especially for scientific formula translation. COBOL, developed around the same time, best supports business applications. BASIC and Pascal were intended to be teaching languages, sometimes making them unwieldy for particular applications.

C, written to develop an entire operating system, is less application- and machine-specific, and is amenable to various kinds of programs. Its skeletal architecture allows programmers—once they've learned the sparse command structure—to add their own routines, commands, and input/output functions called *libraries* (standard C libraries are also available commercially). It's really like the construction of an onion: You have a tiny seed in the center and layer upon layer covering it up. And the fact that input/output is extrinsic to the C language makes for greater portability.

So even though the language itself is small, you can end up with very large programs if you don't economize on your use of libraries, warns Tom Hospelhorn, senior analyst at Mindscape Software. "Sometimes you have very small C programs compiling to what seems to be a very large object," he says. "But what that's usually caused by is you've included a standard library of input/output functions with your object code even though your program may not actually use those." So if you're serious about keeping your finished C programs small, you want to avoid the automatic inclusion of a standard library. You can exclude any unused functions.

C's affinity with machine language gives the user greater programming power, but can also create big problems for novice programmers. Some programmers suggest that C is best not attempted by

A Sample Of C

Here is an example program which prints the numbers from one to ten and describes each number as odd or even. Program 1 is written in Amiga BASIC. By comparing it to Program 2, the C version, you can get a sense of some of the differences between the two languages. For one thing, C makes liberal use of braces—{ }—to group statements into a compound statement, or block, which the language treats as a single statement. Functions are called by supplying the name of the function, with arguments in parentheses—the entire program is a function called `main()`. Variables are declared before use, and comments are framed with `/*` and `*/` characters.

Program 3 is C in a somewhat more condensed, but less easily visualized format. The IF-ELSE construct has been collapsed into one line using the conditional operator (`?:`) to designate the alternative results of the test. This brief sample, however, can't do justice to the qualities which make C an increasingly popular language. There are several excellent texts on C for beginners, available in most bookstores, which will give you a sense of the language's flexibility, power, and efficiency.

Program 1:

```
REM This is a demo program written in AmigaBASIC

PRINT "The numbers from 1 to 10:"

FOR count = 1 TO 10
  PRINT count;" ";
  IF (count AND 1) = 1 THEN PRINT "odd" ELSE PRINT "even"
NEXT count
```

Program 2:

```
/* This is a simple demonstration of a C program,
   written for the Lattice C compiler on the Amiga
   computer */

main()
{
  int count;

  printf("The numbers from 1 to 10:\n");

  for (count = 1; count <= 10; count++)
  {
    printf(" %d ", count);

    if (count & 1 == 1)
      printf("odd\n");
    else
      printf("even\n");
  }
}
```

Program 3:

```
/* A more compact C version of the demo program */
main()
{
  int count;

  printf("The numbers from 1 to 10:\n");
  for (count = 0; ++count <= 10;
       printf(" %d %s\n", count, count & 1 ? "odd" : "even"));
}
```


THE CMO ADVANTAGE

TO ORDER
CALL TOLL FREE
1-800-233-8950
 DEPARTMENT A207

OR MAIL YOUR ORDER TO:

COMPUTER MAIL ORDER
 Department A207
 477 E. Third Street
 Williamsport, PA 17701



POLICY

Add 3% (Minimum \$7.00) shipping and handling. Larger shipments may require additional charges. Personal and company checks require 3 weeks to clear. For faster delivery use your credit card or send cashier's check or bank money order. Pennsylvania residents add 6% sales tax. All prices are subject to change and all items are subject to availability. Defective software will be replaced with the same item only. Hardware will be repaired or replaced at our discretion within the terms and limits of the manufacturer's warranty. We cannot guarantee compatibility. All sales are final and returned shipments are subject to a restocking fee.

EDUCATIONAL INSTITUTIONS

CALL TOLL FREE
1-800-221-4283

CUSTOMER SERVICE
& TECHNICAL SUPPORT
 1-717-327-1450

CANADIAN ORDERS

1-800-268-3974
 Ontario/Quebec

1-416-828-0866
 In Toronto

1-800-268-4559
 Other Provinces

TELEX: 06-218960

2505 Dunwin Drive,
 Mississauga, Ontario
 Canada L5L1T1

All prices shown are for U.S.A. orders. Call the Canadian Office for Canadian prices.

THE CMO ADVANTAGE

- ✓ Next day shipping on all in-stock items.
- ✓ Free easy access order inquiry.
- ✓ Orders from outside Pennsylvania save state sales tax.
- ✓ Free technical support from our factory trained technicians.
- ✓ There is no limit and no deposit on C.O.D. orders.
- ✓ There is no extra charge for using your Visa or MasterCard and your card is not charged until we ship.
- ✓ No waiting period for cashier's checks.
- ✓ We accept purchase orders from qualified corporations. Subject to approval.
- ✓ Educational discounts available to qualified institutions. (See the toll free educational phone number above.)
- ✓ FREE CATALOG MEMBERSHIP

HOME COMPUTERS

ATARI

65XE (64K).....\$89.99
 130XE (128K).....\$139.00
 520ST (512K).....\$369.00
520ST Monochrome System

- 520ST with modulator
 - disk drive
 - mouse
 - logo
 - Basic
 - 1st Word
 - monochrome monitor
- LOW, LOW SYSTEM PRICE**
\$649.00

520ST Color System

- 520ST with modulator
 - disk drive
 - mouse
 - logo
 - Basic
 - 1st Word
 - color monitor
- LOW, LOW SYSTEM PRICE**
\$799.00

800XL 64K.....CALL
 1010 Recorder.....\$49.99
 1050 Disk Drive.....\$149.00
 1020 Printer.....\$29.99
 1027 Letter Quality Printer.....\$129.00
 1030 Direct Connect Modem.....\$59.99
 Comrex 220 Atari.....\$89.99

APPLE

APPLE IIe.....CALL
 APPLE IIc.....CALL
 IIc LCD Display.....\$329.00

COMMODORE

Amiga Package
 • 512K • 2 Drive
 • RGB Monitor.....\$1599.00
C64 Package
 • C64 • C1541
 • Taxan 220.....\$499.00
C128 Package
 • C128 • C1571
 • NAP8562 Monitor.....\$779.00
C128 Computer.....\$269.00
C1571 (Disk Drive for C128).....\$249.00
C1902 (RGB 13" Monitor for C128).....CALL
C1670 (Modem for C128)\$179.00
 C1530 Datasette.....\$39.99
 C1660 Auto Modem.....\$59.99
 DPS 1101 Daisy Printer.....\$339.00
 Comrex 220 (C64 Interface).....\$89.99
 Xetec SuperGraphix 8K.....\$69.99

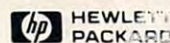
PORTABLE COMPUTERS

NEC

PC-8401 LS.....\$699.00
 PC-8201 Portable Computer.....\$339.00
 PC-8231 Disk Drive.....\$599.00
 PC-8221A Thermal Printers.....\$149.00
 PC-8281A Data Recorder.....\$99.99
 PC-8201-06 8K RAM.....\$79.99

SHARP

PC-1350.....\$149.00
 PC-1261.....\$149.00
 PC-1500A.....\$169.00
 PC-1250A.....\$89.99
 CE-125 Printer/Cassette.....\$129.00
 CE-150 Color Printer Cassette.....\$149.00
 CE-161 16K RAM.....\$129.00



41CV.....\$139.00
 41CX.....\$199.00
 HP 11C.....\$49.99
 HP 12C.....\$75.99
 HP 15C.....\$75.99
 HP 16C.....\$89.99
 HPIL Module.....\$98.99
 HPIL Cassette or Printer.....\$359.99
 Card Reader.....\$143.99
 Extended Function Module.....\$63.99
 Time Module.....\$63.99

We stock the full line of HP calculator products

ACCESSORIES

AMARAY

80 Column Printer Stand.....\$14.99

CURTIS

Side Mount SS-1.....\$19.99
 Side Mount AT SS-2.....\$34.99
 Universal Stand SS-3.....\$19.99
 Diamond SP-1.....\$29.99
 Emerald SP-2.....\$39.99
 Sapphire SPF-1.....\$49.99
 Ruby SPF-2.....\$59.99

DATA SHIELD

300 Watt Backup.....\$379.00
 500 Watt Backup.....\$589.00
 Turbo 350 Watt Backup.....\$449.00
 P125 Power Director.....\$99.99
 P150 Power Director w/Modem\$119.00

KENSINGTON

Master Piece.....\$99.99
 Master Piece +.....\$119.00

KEYTRONICS

KB5150/KB5151/KB5151Jr.....CALL
 KB5152B/KB5153/KB5149Jr.....CALL

MEMORY CHIPS

4164 RAM Chips.....(ea.) \$1.99
 128 RAM Chips.....(ea.) \$12.99
 256 RAM Chips.....(ea.) \$10.99

Polaroid

Palette.....\$1399.00
 Power Processor.....\$229.00
 Illuminated Slide Mounter.....\$39.99
 Polacolor 2 Pack film.....\$18.99

DISKETTES

Dennison

Elephant 5 1/4" SS/SD.....\$9.99
 Elephant 5 1/4" SS/DD.....\$11.99
 Elephant 5 1/4" DS/DD.....\$14.99
 Elephant Premium DS/DD(50).....\$69.99
 Elephant 3 1/2" SS/DD.....\$24.99

IBM

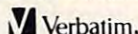
5 1/4" DS/DD floppy disks
 (Box of 10).....\$26.99

GENERIC

DS/DD w/Flip'n File 10.....\$11.99

maxell

3 1/2" SS/DD (10).....\$18.99
 3 1/2" 5 pack SS/DD/Case.....\$9.99
 3 1/2" DS/DD (10).....\$29.99
 5 1/4" MD-1 SS/SD (10).....\$11.99
 5 1/4" MD-2 DS/DD (10).....\$16.99
 5 1/4" MD-2-HD for AT (10).....\$29.99



5 1/4" SS/DD.....\$12.99
 5 1/4" DS/DD.....\$24.99
 Disk Analyzer.....\$24.99

DISK HOLDERS

AMARAY

50 Disk Tub 5 1/4".....\$9.99
 30 Disk Tub 3 1/2".....\$9.99

INNOVATIVE CONCEPTS

Flip'n File 10.....\$2.49
 Flip'n File 50.....\$14.99
 Flip'n File 50 w/lock.....\$19.99
 Flip'n File Data Case.....\$9.99

MODEMS

ANCHOR

Volksmodem.....\$59.99
 Volksmodem 300/1200.....\$189.00
 Signalman Express.....\$259.00
 Lightning 2400 Baud.....\$399.00
 Expressi (PC Halfcard).....\$189.00
 6470 (64/128) 300/1200 Baud.....\$139.00

AST

Reach 1200 Baud Half Card.....\$399.00

DIGITAL DEVICES

AT300 - 300 Baud (Atari).....\$99.99

EVEREX

1200 Baud Internal (IBM/PC).....\$179.00



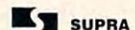
Smartmodem 300.....\$139.00
 Smartmodem 1200.....\$389.00
 Smartmodem 1200B.....\$359.00
 Smartmodem 2400.....\$599.00
 Micromodem IIe.....\$149.00
 Smart Com II.....\$89.99
 Chronograph.....\$199.00
 Transet 1000.....\$309.00



Smart Cat Plus.....\$299.00
 J-Cat.....\$99.99
 Novation 2400.....\$499.00
 Apple Cat II.....\$219.00
 212 Apple Cat II.....\$379.00
 Apple Cat 212 Upgrade.....\$229.00



Quadmodem II
 300/1200.....\$339.00
 300/1200/2400.....\$499.00



MPP-1064 AD/AA (C-64).....\$69.99

DRIVES

HARD

CORE

AT20-AT72MB.....CALL

EVEREX

60 Meg Internal Backup System\$799.00

IOMEGA

A110H Single 10.....CALL
 A210H 10 + 10.....CALL
 A120H Single 20.....CALL
 A220H 20 + 20.....CALL
 Save on 10 & 20 Cartr.....CALL

IRWIN

Tape Backup.....CALL

KITS

10 Meg with controller.....\$399.00

PRIAM

40, 60 MB Inner Space.....CALL
 Shared Data.....CALL
 Shared Space.....CALL

TALLGRASS

25, 35, 50, 80 meg (PC)
from \$1299.00

FLOPPY

ALLIED TECHNOLOGY

Apple II, II+ , IIe 1/2 height.....\$109.00

INDUS

Atari GT.....\$199.00
 C-64 /128 GT.....\$199.00

MSD

SD1 C-64 Single.....\$219.00
 SD2 C-64 Dual.....\$469.00

TANDON

320K 5 1/4" (PC).....\$119.00

TEAC

320K 5 1/4".....\$119.00

DEPT. A207

CALL TOLL-FREE

www.commodore.ca

SOFTWARE FOR IBM

PRINTERS

MULTIFUNCTION CARDS

IBM

ANSA SOFTWARE

Paradox.....\$499.00

ASHTON-TATE

Framework II.....\$389.00

dBase III Plus.....\$389.00

BATTERIES INCLUDED

Isgur Portfolio.....\$159.00

BORLAND

Lightening.....\$59.99

Sidekick (unprotected).....\$57.99

Reflex.....\$59.99

Travelling Sidekick.....\$44.99

Turbo Prolog.....\$64.99

CENTRAL POINT

Copy II PC-Backup.....\$29.99

PC Option Board.....\$84.99

DECISION RESOURCES

Chartmaster.....\$229.00

Signmaster.....\$159.00

Diagram Master.....\$209.00

FIFTH GENERATION

Fast Back.....\$99.99

FUNK SOFTWARE

Sideways.....\$44.99

HARVARD SOFTWARE INC.

Total Project Manager.....\$269.00

Presentation Graphics.....\$239.00

LIFETREE

Volkswriter III.....\$159.00

LIVING VIDEOTEK

Think Tank.....\$109.00

Ready.....\$64.99

LOTUS

Symphony.....CALL

1-2-3 Version 2.....CALL

MECA SOFTWARE

Managing Your Money 2.0.....\$99.99

Manage Your Market.....\$89.99

MICROPRO

Easy.....\$94.99

WordStar 2000.....\$239.00

WordStar 2000+.....\$289.00

WordStar Professional.....\$189.00

MICRORIM SOFTWARE

R:Base 4000.....\$249.00

R:Base 5000.....\$359.00

Clout 2.0.....\$129.00

MICROSOFT

Flight Simulator.....\$34.99

MultiPlan.....\$129.00

Word.....\$249.00

Mouse.....\$139.00

MICROSTUF

Crosstalk XVI.....\$89.99

Crosstalk Mark IV.....\$149.00

Remote.....\$89.99

MULTIMATE

Multi Mate Word Proc.....\$219.00

Advantage.....\$289.00

On File.....\$89.99

Just Write.....\$89.99

NOUNEMON

Intuit.....\$69.99

NORTON

Norton Utilities 3.1.....\$57.99

ONE STEP

Golf's Best.....\$34.99

PFS: IBM

Proof.....\$59.99

File/Graph.....(ea.)\$84.99

Report.....\$74.99

Write/Proof Combo.....\$84.99

PROFESSIONAL SOFTWARE

Write-N-Spell.....\$89.99

THE SOFTWARE GROUP

Enable.....\$329.00

SATELLITE SYSTEMS

Word Perfect 4.1.....\$219.00

SORCIM/IUS

Accounting.....\$299.00

API/AR/GL/INV/OE.....(ea.)\$299.00

SuperCalc III.....\$199.00

EasyWriter II System.....\$239.00

Super Project.....\$199.00

SUBLOGIC

Jet.....\$37.99

Canon

A40,A50,A55.....CALL

LBP-8A Laser.....CALL

CITIZEN

MSP-10 (80 col.).....\$279.00

MSP-15 (132 col.).....\$389.00

MSP-20 (80 col.).....\$349.00

MSP-25 (132 col.).....\$509.00

C. ITOH

Prowriter 7500.....\$169.00

Prowriter 1550P.....\$349.00

Starwriter 10-30.....\$399.00

3500 Tri Printer.....\$1499.00

corona

Lazer LP-300.....\$2799.00

DIABLO

620 Daisywheel.....\$299.00

D25 Daisywheel.....\$549.00

635 Daisywheel.....\$1099.00

D80IF Daisywheel.....CALL

daisywriter

2000.....\$699.00

EPSON

Homewriter 10, LX-80.....CALL

FX-85, FX-286, RX-100.....CALL

DX-10, DX-20, DX-35.....CALL

SQ-2000, Hi-80, HS-80, AP-80.....CALL

LQ-800, LQ-1000.....CALL

JUKI

6000 Letter Quality.....CALL

6100 Letter Quality.....CALL

6200 Letter Quality.....CALL

6300 Letter Quality.....CALL

6500 Letter Quality.....CALL

5510 Dot Matrix Color.....CALL

LEGEND

808 Dot Matrix 100 cps.....\$179.00

1080 Dot Matrix 100 cps.....\$259.00

1380 Dot Matrix 130 cps.....\$289.00

1385 Dot Matrix 165 cps.....\$339.00

NEC

3000 Series.....\$779.00

8000 Series.....\$1099.00

ELF 360.....\$399.00

Pinwriter 560.....\$999.00

OKIDATA

182, 183, 192, 193, 2410, 84.....CALL

Okimate 10 (Specify C64/Atari)\$189.00

Okimate 20 (IBM).....CALL

Panasonic

KX1080.....NEW

KX1091.....\$259.00

KX1092.....\$389.00

KX1592.....\$469.00

KX1595.....\$659.00

QUADRAM

Quadjet.....\$399.00

Quad Laser.....CALL

SILVER-REED

500 Letter Quality.....\$219.00

550 Letter Quality.....\$419.00

800 Letter Quality.....\$699.00

star

SG-10C (C64 Interface).....CALL

SB/SD/SG/SR Series.....CALL

Powertype Letter Quality.....CALL

Texas Instruments

TI850.....\$529.00

TI855.....\$639.00

TI865.....\$799.00

TOSHIBA

P321 (80 column).....\$499.00

P341 (132 column).....\$799.00

P351 (132 column).....\$1049.00

AST

RamVantage.....\$349.00

Rampage-PC.....\$379.00

Rampage-AT.....CALL

Six Pack Plus.....\$229.00

I/O Plus II.....\$139.00

Advantage-AT.....\$399.00

Preview Mono.....\$299.00

PC Net Cards.....\$379.00

5251/11 On-line.....\$669.00

5251/12 Remote.....\$579.00

dca

IRMA 3270.....\$879.00

IRMA Print.....\$999.00

IRMA Smart Alec.....\$779.00

EVEREX

Edge Card.....\$259.00

Graphics Edge.....\$219.00

Magic Card II.....\$159.00

Magic Card I.....\$99.99

HERCULES

Graphics.....\$299.00

Color.....\$159.00

IBM Associates

IDEA 5251.....\$549.00

INTEL

PCNC8087 5MHz.....CALL

PCNC8087-2 8 MHz.....CALL

PCNC80287 6 MHz.....FOR

1010 PC-Above Board.....YOUR

1110 PS-Above Board.....PC

2010 AT-Above Board.....CALL

MYLEX

The Chairman.....\$439.00

PARADISE

Color/Mono Card.....\$149.00

Modular Graphics Card.....\$199.00

Multi Display Card.....\$199.00

Five Pack C, S, O-384K.....\$99.99

High Res Mono.....\$169.00

PERSYST

Bob Board.....\$359.00

QUADRAM

Quadport-AT.....\$119.00

Liberty-AT (128K).....\$349.00

The Gold Quadboard.....\$449.00

The Silver Quadboard.....\$239.00

Expanded Quadboard.....\$199.00

Liberty.....\$309.00

QuadPrint.....\$499.00

QuadLink.....\$399.00

QuadColor.....\$199.00

Quadboard-AT.....\$399.00

8600 E.G.A. card.....\$399.00

STB

EGA Plus.....\$379.00

TECMAR

Captain - 64.....\$199.00

Graphics Master.....\$469.00

VIDEO-7

EGA.....\$429.00

INTERFACES

AST

Multi I/O (Apple II).....\$149.00

DIGITAL DEVICES

Ape Face (Atari).....\$49.99

U-Print A (Atari).....\$54.99

U-A16/Buffer (Atari).....\$74.99

U-Call Interface (Atari).....\$39.99

U-Print C (C64).....\$49.99

P-16 Print Buffer.....\$74.99

U-Print 16 apple IIc.....\$89.99

MICRO R & D

Apple IIc Parallel.....\$49.99

Kaypro 2000 Parallel.....\$49.99

C64/128.....\$59.99

Orange Mikro

Grappier CD (C64).....\$89.99

Grappier Plus (Ile, IIc).....\$89.99

Grappier C (Ile).....\$89.99

Grappier 16K (Ile, II+).....\$139.00

PRACTICAL PERIPHERALS

Graphicard.....\$69.99

Serial Card.....\$99.99

Microbuffer II+ 64K.....\$169.00

QUADRAM

Microfazer.....from \$139.00

Efazer (Epson).....from \$79.99

IBM PC SYSTEMS

Configured to your specifications.

Call for Best Price!

IBM-PC, IBM-XT, IBM-AT

Safari (7300).....CALL
6300.....CALL

corona

PPC400 Dual Portable.....\$1289.00

PPCXT 10 meg Portable.....\$1989.00

PC40022 Dual Desktop.....\$1389.00

PC400-HD2 10 meg.....\$1989.00

ITT

ITT X-TRA.....CALL

256K, 2 Drive System.....CALL

256K, 10 meg Hard Drive System.....CALL

XP5, 20 meg.....CALL

KAYPRO

KP-2000 Portable.....CALL

Kaypro PC.....CALL

SANYO

MBC 550-2, MBC 555-2, MBC 675 Por-

table, MBC775, MBC 880 DesktopCALL

SPERRY

Sperry-AT.....as low as \$1749.00

Sperry-IT.....as low as \$2699.00

Call for Specific Configuration!

All Models.....CALL

ZENITH

PC-138 Series, PC-148 Series, PC-158

Series, PC-160 Series, PC-171 Series,

AT-200 Series.....CALL

MONITORS

AMDEK

Video 300 Green.....\$119.00

Video 300A Amber.....\$129.00

Video 310A Amber TTL.....\$159.00

Color 600 Hi-Res. RGB.....\$399.00

Color 722 Dual Mode.....\$529.00

Color 725.....CALL

Color 730.....CALL

MAGNAVOX

8562 RGB/Composite.....\$279.00

613 TTL Green.....\$99.99

623 TTL Amber.....\$99.99

NEC

JB1205A.....\$79.99

JB1270G/1275A.....(ea.)\$99.99

JB1280G TTL Green.....\$129.00

JB1285A TTL Amber.....\$129.00

JC1401 Multi Sync RGB.....CALL

PRINCETON

MAX-12 Amber.....\$179.00

HX-9 9" RGB.....\$469.00

HX-9E Enhanced.....\$519.00

HX-12 12" RGB.....\$469.00

HX-12E Enhanced.....\$559.00

SR-12 Hi-Res.....\$599.00

All the exciting, entertaining, and educational games, applications, and utilities from **COMPUTE!** magazine are now available on disk

**for your Commodore,
Atari, Apple, or IBM
personal computer.**

The COMPUTE! Disk

A new *COMPUTE! Disk* is published every month, rotating among the four major machines covered by *COMPUTE!*: Commodore 64 and 128; Atari 400/800, XL, and XE; Apple II-series; and IBM PC, PCjr, and compatibles.

Every three months you can receive a disk with all the quality programs from the previous three issues of *COMPUTE!* that will run on your brand of computer.

Like the popular *COMPUTE!'s Gazette Disk*, the *COMPUTE! Disk* is ready-to-load and error-free. It saves you valuable hours of typing time and eliminates typing errors.

With a subscription, you will receive one disk every three months for a total of four disks a year—for only \$39.95. That saves you \$20 a year off the single-issue cost.

Or you can order individual issues of the *Disk* for \$12.95 a disk plus \$2.00 shipping and handling.

Remember to specify your type of computer when ordering the *COMPUTE! Disk*. You'll find more information about this month's *COMPUTE! Disk* in this issue. (Note: You'll need the corresponding issues of *COMPUTE!* magazine to use the *Disk* since the disk will have no documentation.)

For fastest service when ordering a subscription to the *COMPUTE! Disk*, call toll free 1-800-247-5470 (in Iowa 1-800-532-1272).

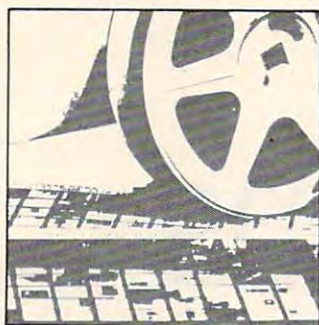
For more details or to order individual issues of the *COMPUTE! Disk*, call our Customer Service Department toll free at 1-800-346-6767 (in New York 212-887-8525).

Please allow 4–6 weeks after placing an order for your first disk to arrive.

COMPUTE! Publications, Inc. 

One of the ABC Publishing Companies
825 7th Avenue, 6th Floor, New York, NY 10019
Publishers of *COMPUTE!*, *COMPUTE!'s Gazette*, *COMPUTE!'s Gazette Disk*, *COMPUTE! Books*, and *COMPUTE!'s Apple Applications*

This Publication is available in Microform.



University Microfilms International

Please send additional information

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

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

people unfamiliar with machine language. At the very least, a disciplined, structured approach to C programming is highly recommended.

"If you haven't done a lot of assembly language first, going from Pascal or BASIC to C is going to be more of a challenge because it is getting closer to the machine," says Jeff Steinwedel, engineering manager at Activision Software. "It's like going from BASIC to assembly language. You have to know a lot more about what's going on internally. The language system is going to be less help in bailing you out of a situation or preventing you from making bugs in the first place. But if you've done a lot of assembly language programming, the transition to C will be fairly straightforward."

One of the characteristics shared by C and machine language is the ability to manipulate memory directly (in C, pointers are used). In higher level languages, memory access is often indirect. And C's heavy reliance on pointers is very powerful in general, says Hospelhorn.

But because you tend to do a lot of things with pointers, and because there's no error checking on pointers to see what they point to, you can wind up creating problems for yourself if you're not a disciplined programmer, he says. "It's possible to damage your program, to set memory locations that you didn't intend to if you're not well aware of where your pointers are pointing. That can make debugging sort of tedious. It's certainly possible for a programmer to generate very difficult errors—more so than some other languages that do a better job of checking things for you."

C, then, isn't a free lunch. What C gives you is a tremendous amount of power and flexibility, a feeling of liberation, he says, because most of the things you can think of, you can do in some way. But with that liberation comes a certain amount of responsibility. "It's always best to plan things out before forging ahead, but it might be a little more true in C."

And there's a big payoff after a C program is successfully completed and debugged: Translating that program to run on another 68000-based computer is easier than transporting a program from, say, an

Apple II to an Atari 800. Part of the reason for this, of course, lies in the memory limitations of the eight-bit machines. "When you're in a very constrained machine without a lot of memory, like the Apple II, you tend to do a lot more in assembly language because you don't have the space," says Steinwedel. "With an Atari 1040 ST with a megabyte of memory, it probably isn't going to hurt you."

"But the real advantage is to be able to produce the code quickly and rapidly transport it from one environment to another. Something running in GEM on an IBM can move very quickly to the ST."

There's one additional reason for the popularity of C on the 68000-based machines. It's often mentioned almost as an afterthought, but it's significant. There are currently many good C compilers available. Other languages aren't as well-supported at this time with efficient, tested compilers. Lattice (Glen Ellyn, IL) and Manx Software Systems (Shrewsbury, NJ) publish C compilers that are enjoying popularity with 68000-based computer programmers, as do several other software publishers. There are also versions of interpreted C available, for programmers who want to debug more easily. Some use interpreters to write the programs and then, after all the problems are ironed out, run the source code through a compiler for the greatest runtime efficiency.

Although the friendly user interfaces of the Apple Macintosh, Atari ST, and Commodore Amiga have enticed nontechnical consumers to purchase them as applications machines, sales of languages and how-to books and other programming utilities indicate that many buyers are planning to write their own programs. The special benefits of C on 68000 machines recommend it to experienced machine language programmers. Likewise, people who have worked with higher level languages and want to move a bit closer to the inner workings of the computer might want to sample this special way of communicating with their machines. A decade and a half after its inception, the C programming language is approaching a new level of popularity. ©

COMPUTE! Subscriber Services

Please help us serve you better. If you need to contact us for any of the reasons listed below, write to us at:

COMPUTE! Magazine
P.O. Box 10954
Des Moines, IA 50340

or call the Toll Free number listed below.

Change Of Address. Please allow us 6-8 weeks to effect the change; send your current mailing label along with your new address.

Renewal. Should you wish to renew your **COMPUTE!** subscription before we remind you to, send your current mailing label with payment or charge number or call the Toll Free number listed below.

New Subscription. A one year (12 month) US subscription to **COMPUTE!** is \$24.00 (2 years, \$45.00; 3 years, \$65.00. For subscription rates outside the US, see staff page). Send us your name and address or call the Toll Free number listed below.

Delivery Problems. If you receive duplicate issues of **COMPUTE!**, if you experience late delivery or if you have problems with your subscription, please call the Toll Free number listed below.

COMPUTE!
1-800-247-5470
In IA **1-800-532-1272**

The Top Five Free Programs For Your Computer

Arlan R. Levitan

Good software doesn't have to be expensive. You can accumulate a respectable software library merely by taking advantage of the thousands of programs in the public domain—that is, programs which are given away free by their authors. Another alternative, the "shareware" concept, lets you test-drive a program for free and make a voluntary contribution if you like. Here's a guide to public domain software and shareware, plus the results of a survey in which users all over the country voted for their top five favorites.

Does the thought of paying more for a program than you laid out for your computer make you grumpy and irascible? Cheer up. There's a wealth of programs available for your computer that cost little or nothing at all. Public domain and shareware programs can provide you with a never-ending supply of grist for your computer's mill.

The idea of public domain software has been around since the early computer hobbyists first started sharing their programs with each other. People would try running each other's programs, suggest improvements, or make the improvements themselves. Few people copyrighted their programs because they were hobbyists rather than software authors trying to make a living. Legally, all it takes to place a program in the public domain is for the author to declare it so. (Of course, this excludes most programs published in magazines and books, which are nearly always copyrighted to protect the authors.)

Public domain programs can be freely exchanged between individuals or distributed by user

groups and computer bulletin board systems (BBSs). They come with no warranties, packaging, or customer support. They are gifts to the public and vary in quality from marginal to very good.

To determine which public domain programs are the most popular among users, in April we conducted a survey over three commercial information services: CompuServe, The Source, and Delphi. Below are the results of this informal survey. For each personal computer, we've listed the top five programs. The type of program is identified within parentheses.

We have excluded from consideration programs that are not truly in the public domain, including programs which elicit a fee for documentation, and programs which have been published, are in widespread use, but are definitely not in the public domain—such as COMPUTE!'s own *SpeedScript*, for example.

You'll notice that many of the popular programs on the list are terminal programs. This is probably due to the fact that the survey was conducted online among telecomputing enthusiasts.

To obtain copies of any of these programs, try contacting your local user group or logging onto a BBS or commercial information ser-

vice. Friends and coworkers are also valuable sources for public domain programs.

Another type of freely distributed software that is sometimes confused with public domain software is *shareware* (also called *user-supported software*). The concept of shareware came about as a response to the negative aspects of marketing software commercially.

It seems that almost everybody likes to complain that software is too expensive. Critics of the software industry claim that prices are inflated by a charge-what-the-market-will-bear attitude as the product filters through distribution channels. The manufacturer typically sells to a distributor, who in turn sells to a retailer. Each middleman adds a markup. The author of the software receives only a small percentage of the selling price.

Critics argue that this practice causes a serious problem: The perception of high prices encourages unauthorized duplication of software. This leads to a classic conflict between the manufacturers and the software pirates. Manufacturers may be tempted to boost their prices to make up for expected losses to piracy, and pirates may justify copying because they say prices are unreasonably high.

For these and other reasons, some software authors decide to market their programs themselves. There have been few success stories among those who've tried this approach. The authors attempt to work within the established marketplace, but usually fail because they lack the resources necessary to promote, advertise, and distribute their product.

About four years ago, a programmer named Andrew Fluegelman wrote a terminal program for IBM computers called *PC-Talk III*. To distribute his program, Fluegelman combined aspects of both public domain and commercial software to come up with a new category he called *Freeware*. Freeware is based on three concepts:

- Before buying a program, computer users should have the opportunity to fully assess its value by

using it extensively to determine whether it serves their needs.

- Original software of high quality written by independent authors will be supported by the personal computing community.

- Copying of these programs should be *encouraged*, rather than discouraged. The ease of disseminating programs outside traditional commercial channels should be exploited by software authors to maximize distribution.

Fluegelman actually trademarked the term *Freeware*, so as these ideas spread and other authors began following suit, the term *shareware* was coined for general use. Here's how shareware typically works:

Anyone can get a copy of a shareware program. Usually, you obtain it from a local user group or BBS. Since there is no packaging or manual, any documentation is generally in the form of a text file on the disk or BBS. You must print out a hardcopy if you want a manual for reference purposes.

Shareware programs contain a notice suggesting that you send a certain contribution to the author if you find the program useful. The contribution is voluntary, and even if none is made, you're encouraged to share the program with others.

Although no shareware authors are reported to be making a killing, many are said to be realizing a steady stream of supplemental income.

How good is shareware? The best of it is quite good indeed, and often better suited to the needs and abilities of casual users than more expensive commercial programs. If you're willing to do without fancy manuals and can rely on fellow users for technical support, shareware may be right for you.

Here are the top five freely distributed programs for each popular personal computer. Shareware programs are denoted with an asterisk (*). You'll notice that only four programs are listed for the Commodore 64/128. That's because the other programs which received votes are not truly in the public domain—including two which are copyrighted by COMPUTE!.

Commodore 64/128

Comm Term (Terminal program)
Haunted Hill (Game)*
Disk Doctor 128 (Utility)
Blue Thunder (Game)

Atari 400/800/XL/XE

AMIS (Bulletin board system)
AMODEM (Terminal program)
MYRIAPEDE (Game)
POKEY Player (Music)
AMENU (Program autoloader)

Atari ST Series

STerminal (Terminal program)
STCalc (Calculator desk accessory)*
Megaroids (Game)
RMDISK (RAM disk utility)
COPY (File utility for single-drive systems)

Apple II Series

EAMON (Adventure game)
FreeWriter (Word processor)*
EVE (Terminal program)*
RAMDISK128 (RAM disk utility)
ABBS (Bulletin board system)

Commodore Amiga

Aterm (Terminal program)
StarTerm (Terminal program)
Mandelbrot (Graphics demo)
Hack (Adventure game)
EMACS (Text editor)

IBM PC/PCjr

MEMBRAIN (RAM disk utility)
PROCOMM (Terminal program)*
PC-File (Database manager)*
RBBS (Bulletin board system)
PC-Write (Word processor)*

Apple Macintosh

Red Ryder (Terminal program)*
BINHEX (File conversion utility)*
MazeWars (Game)
VMCO (Vocal/visual terminal program)
ResEdit (Resource editor)

Texas Instruments TI-99/4A

Fast-Term (Terminal program)
Disk Manager 1000 (Disk cataloger)
FUNL Writer (Word processor)
NeatList (Utility)
MassCopy (Utility)

©



Hex War

Todd Heimarck, Assistant Editor

You float high above a distant planet, controlling robot armies below. Can you take control of the priceless mining turf planetside, or will your opponent's robot crews prevail? To win at this thoughtfully designed, engaging strategy game, you'll need foresight and conceptual skills rather than a quick hand on the joystick. The original version is written for the Commodore 128. We've programmed new versions for the Commodore 64, Apple II series, IBM PC/PCjr, Atari 400/800/XL/XE, and the Amiga. A joystick is required to play the Commodore 128 and 64 versions. The IBM PC/PCjr version requires Cartridge BASIC for the PCjr and BASICA plus a color/graphics adapter for the PC. The Atari version requires a joystick and at least 48K of memory. The Amiga version requires 512K.

"Hex War" is a two-player strategy game that can be played five different ways, and there are limitless variations. But the basic premise is always the same: You and an opponent move armies on a field of hexagons, attempting to capture territory.

The goal of the first two games is simple: capture the capital city of the other player. In game 1, the capital cities are far apart; you must devote some of your armies to defending your own capital while attempting to breach the walls of the other capital. Game 2 puts the capitals near each other, so offense and defense tend to merge in this scenario. Most of the action takes place within a small area of the battlefield.

Games 3 and 4 spread the action over a wider area. In the third game, your object is to occupy eight

of the twelve cities on the game board. Six cities occupy the periphery, and six are in the center of the playfield. Game 4 requires actual control of six cities; you must have an army in the city, one that's not involved in a battle, before you're credited with control (this version will probably take the most amount of time to play).

Although the first four scenarios encourage a commitment to battle, you employ different tactics in the fifth. The goal here is to acquire 40 of the 61 hexes, so you need some free armies to move around. As soon as you claim 40 hexes, you win the game.

Typing It In

Hex War is written in BASIC, with some important information in DATA statements. Type in the version for your computer and be sure

to save a copy. Refer to the notes below for special instructions specific to your computer. After the game has been saved, type RUN to begin playing.

When you first run Hex War, the computer pauses to set up the screen, then displays a menu of five choices. The five different games are explained in detail below. If you're new to the game, press the 1 key to choose game 1. There will be another short pause while the variables are initialized, and then you'll see a playfield with 61 hex shapes, containing four armies on each side (see photos).

Hexes And Hexadecimal

A chess board has 64 squares arranged in a rectilinear grid. Hex War gives you a playing field of 61 hexagons (almost as many as a chess board), but they're part of a six-sided honeycomb field. If you've played war games before, you may recognize the hexes.

If you're using a Commodore 128, 64, or Atari computer, plug in the joystick before playing (use port 2 for the 128 and 64). The other versions use keyboard controls as explained below. At first, the cursor movement may seem unusual. The cursor travels not up-down/left-right, but northeast-southeast/northwest-southwest. To make the movement less confusing, turn your joystick 45 degrees clockwise, so that what was up becomes northeast, and so on.

Each hex has six neighbors, so an army can move in six possible directions. To travel left and right, you'll have to push the joystick twice (for example, up and right on the joystick to move one hex to the right, which counts as one movement).

Army strengths are listed in hexadecimal (base 16) numbers, so the four armies labeled 40 actually have strengths of 64 (the hexadecimal value 40 equals 64 in our everyday decimal numbering system). At the beginning of a turn, any army has exactly three movement points. It requires one point to move an army into a neutral or enemy-controlled zone. To move *through* the same zone also requires a point. To move into and through a friendly hex requires a total of one

point. This means you can move a single army through two neutral or enemy hexes in any one turn, but the same army can move through up to three friendly zones during a turn.

Select an army by moving the cursor onto it. Click the joystick button once, then position the cursor on a neighboring hex and click again. If you wish to stop, click again, and two plus signs (++) will appear, signaling that no more movement can occur. Otherwise, position the cursor on another neighboring hex and click.

Zones Of Control

Each army controls the six contiguous hexes surrounding its resident hex. If you enter an enemy's zone of control, you forfeit any additional moves and must prepare for battle. In addition, an army that begins the turn in a zone of control cannot move until the battle is resolved.

Robots Vs. Robots

In this game, you aren't really on the planet, but parked high above it in a remote mothership. You've landed some robots to explore the area, and they've encountered robots belonging to another explorer. Your robots, or *bots* as you call them, follow your orders to advance toward the other bots. Each bot has a mining laser which can stop or disable the other bots. Also, your bots have disruptor beams which can daze another bot, temporarily confusing it. When two bot-groups come close to each other, they shoot lasers and disruptors until one army of bots is disabled.

Three things can happen to a robot which suffers a hit. If the robot suffers a direct hit in its logic unit by a laser, it is vaporized. It is destroyed forever and never reappears in play.

The second thing that can happen is injury. If the laser beam is deflected, the robot is out of commission until it can be transported back to a botspital. An injured bot is frozen in place until the battle is finished, after which the victorious army carts away the injured bots to be repaired and reused.

Thus, winning a battle means you evacuate both the friendly injured and the enemy injured. After

all of the injured bots recover, *they join the force in whose botspital they were healed*. In effect, injured bots eventually become members of the army which won the battle in which they were damaged.

The third possibility is confusion: The robot is temporarily disoriented for two turns. When the time has passed, the robot is ready again.

Reprogramming Bots

Moving the cursor onto an army of robots brings up a status window in the upper-left corner of the screen. The number in reverse video is unimportant; it's the army number (which may change as the game progresses).

The four numbers underneath are significant, however. The first is the army's active strength (in decimal). The second is the number of injured robots, which will be transported to the botspital of whichever side wins the battle. The third—on the line below—is the number of disrupted robots who will be available for combat in the next turn. The fourth number is how many robots can join the active force two turns from now.

If one side is able to reduce the other player's active force to zero, two things happen. The winner sends all injured bots away to be repaired. The winning side also collects all enemy bots (injured or dazed) and sends them to the reinforcement center to be reprogrammed. Eventually all these bots will be available to the winner of this particular battle for future engagements.

Reinforcements And Mergers

At the start of the game, you'll see some armies positioned outside of the hex field. These are reinforcements and reserves in transit to the battle. Player one's reinforcements enter at the bottom right corner; player two's enter at the top left. The line of new armies moves counterclockwise; the army next to the entry point is the next to enter the battlefield.

However, the reinforcements cannot enter the battlefield if an army (friendly or enemy) is blocking their way. Keep your armies off

Lyco Computer Marketing & Consultants



ATARI

| | |
|---------------|------|
| 130XE | CALL |
| 65XE | CALL |
| 800XL | CALL |
| 520ST | CALL |
| 1050 Drive | 145 |
| 1027 Printer | 145 |
| 850 Interface | 109 |
| SF314 Drive | 229 |
| SF354 Drive | 179 |

| | |
|---------------|------|
| 1040 St (New) | Call |
| XM301 Modem | 31 |

BRODERBUND (Atari)

| | |
|----------------------|-------|
| The Print Shop | 28.95 |
| Graphics Library | 18.95 |
| Graphics Library II | 19.50 |
| Graphics Library III | 19.50 |
| Bank St. Writer | 42.75 |
| Whistler's Brother | 18.95 |
| Spelunker | 18.95 |
| Stealth | 18.95 |
| Serpent's Star | 24.95 |
| Mask of the Sun | 24.95 |

MICROLEAGUE (Atari)

| | |
|-----------|-------|
| Baseball | 24.95 |
| GM disk | 24.95 |
| Team disk | 14.95 |

SSI (Atari)

| | |
|-----------|-------|
| Nam | 24.75 |
| Mechraged | 34.95 |
| Antietam | 29.95 |
| USAF | 34.95 |

ACTIVISION (520 St)

| | |
|---------------|-------|
| Borrowed Time | 29.75 |
| Music Studio | 29.75 |
| Hackler | 26.75 |
| Mindshadow | 29.75 |

VIP TECH

| | |
|------------------------|-----|
| VIP Professional 520ST | 115 |
| VIP LITE 520ST | 65 |
| VIP Professional Amiga | 129 |

MARK OF UNICORN (520ST)

| | |
|--------------|--------|
| HEX | 29.95 |
| MINCE | 129.95 |
| PC/InterComm | 99.95 |

HABA (520 St)

| | |
|--------|-------|
| Writer | 49.95 |
|--------|-------|

QUICKVIEW (520 St)

| | |
|-----------|-------|
| Zoomracks | 49.95 |
|-----------|-------|

ACTIVISION (Atari)

| | |
|----------------|-------|
| Hackler | 15.75 |
| Mindshadow | 15.75 |
| Ghostbusters | 15.75 |
| Great Am. Race | 15.75 |
| Music Studio | 20.75 |
| Space Shuttle | 15.75 |

MICROPROSE (Atari)

| | |
|--------------------|-------|
| Kennedy Approach | 21.75 |
| Crusade in Europe | 24.75 |
| Decision in Desert | 24.75 |
| Solo Flight | 20.75 |
| Nato Commander | 20.75 |
| Spitfire Ace | 18.75 |
| F-15 Strike Eagle | 20.75 |
| Silent Service | 20.75 |
| Conflict in Nam | 24.75 |

SYNAPSE (Atari)

| | |
|-----------|-------|
| Synfile | 29.95 |
| Syncalc | 29.95 |
| Template | 14.95 |
| Mindwheel | 24.75 |
| Brimstone | 24.75 |

WICO Joysticks

| | |
|---------------------|-------|
| 15-9714 Bat Handle | 16.75 |
| 50-2030 Boss | 11.99 |
| 50-2002 Super 3-Way | 19.99 |

REDIFORM PAPER

| | |
|-------------------------------|-------|
| Qty 1000 9 1/2x11 white laser | 19.95 |
| Qty 3000 9 1/2x11 white laser | 29.95 |
| Qty 1000 9 1/2x11 white laser | 16.95 |
| Qty 200 9 1/2x11 white laser | 5.95 |
| Qty 1000 Mailing labels 1x3 | 6.95 |

SUB LOGIC (Atari)

| | |
|-----------------------|-------|
| Flight Simulator II | 32.75 |
| Night Mission Pinball | 20.75 |



ACTIVISION (Apple)

| | |
|---------------|-------|
| Alter Ego | 28.75 |
| Little People | 24.75 |
| Mindshadow | 24.75 |
| Hackler | 24.75 |
| Gamemaker | 24.75 |

BRODERBUND (Apple)

| | |
|----------------------|-------|
| The Print Shop | 31.50 |
| Graphic Librarys EA | 18.50 |
| Bank St. Writer 128K | 42.75 |
| Bank St. Speller | 42.75 |
| Carmen Sandiego | 22.75 |
| Karateka | 22.75 |
| Captain Goodnight | 22.75 |
| Muppet Cruise | 25.75 |
| P. S. Companion | 24.75 |
| Science Kit | 35.95 |

MICROPROSE (Apple)

| | |
|--------------------|-------|
| Crusade in Europe | 24.75 |
| Decision in Desert | 24.75 |
| F-15 Strike Eagle | 20.75 |
| NATO Commander | 20.75 |
| Silent Service | 20.75 |
| Solo Flight | 20.75 |

SSI (Apple)

| | |
|-------------------|-------|
| Phantasia II | 24.75 |
| Wizard's Crown | 24.75 |
| Rings of Zilfin | 24.75 |
| Colonial Conquest | 24.75 |
| Battlegroup | 35.75 |
| NAM | 29.75 |

MICROLEAGUE (Apple)

| | |
|--------------|-------|
| M L Baseball | 24.95 |
| General Mgr | 24.95 |



COMMODORE

| | |
|-----------------|------|
| 128 | CALL |
| C 1571 Drive | CALL |
| C 1902-A | CALL |
| C-64 | CALL |
| C 1541 Drive | CALL |
| C 1670 Modem | CALL |
| C-64 Computer | CALL |
| MPS801 Printer | LOW |
| C 1350 Mouse | 42 |
| C 1700 128K RAM | 145 |
| C 1750 512K RAM | 269 |
| JANE | 35 |

ACTIVISION (C-64/128)

| | |
|---------------|-------|
| Alter Ego | 28.75 |
| Hackler | 18.75 |
| Little People | 20.75 |
| Gamemaker | 24.75 |
| Borrowed Time | 18.75 |
| Space Shuttle | 18.75 |
| Music Studio | 24.75 |
| Mindshadow | 18.75 |
| Roadrace | 18.75 |
| Fast Tracks | 22.75 |
| Count Down | 18.75 |
| Ghostbusters | 22.75 |

SUB LOGIC (C-64)

| | |
|-----------------------|-------|
| Flight Simulator II | 32.75 |
| Night Mission Pinball | 20.75 |

INNOVATIVE CONCEPTS

| | |
|---------------------|-------|
| Flip-n-File 10 | 3.50 |
| Flip-n-File 15 | 8.25 |
| Flip-n-File 25 Lock | 17.95 |
| Flip-n-File 50 | 17.25 |
| Flip-n-File 50 Lock | 22.95 |
| Flip-n-File Rom | 17.25 |

PERSONAL PERIPHERALS

| | |
|----------------------|-------|
| Super Sketch C-64 | 29.95 |
| Printer Utility C-64 | 14.99 |

CARDCO

| | |
|------------------------|-------|
| Numeric Keypad | 34.95 |
| CB/5 5-slot Board (64) | 49.95 |
| CB/2 2-slot Board (64) | 21.95 |
| S More Basic Rom | 49.95 |
| Mail Now-64 | 24.95 |
| Spell Now-64 | 24.95 |
| File Now-64 | 24.95 |
| Paint Now-64 | 24.95 |
| Calc Now-64 | 24.95 |
| Super Printer Utility | 24.95 |

MICROPROSE (C-64)

| | |
|--------------------|-------|
| Kennedy Approach | 21.75 |
| Crusade in Europe | 24.75 |
| Decision in Desert | 24.75 |
| Solo Flight | 20.75 |
| Nato Commander | 20.75 |
| Spitfire Ace | 18.75 |
| F-15 Strike Eagle | 20.75 |
| Acrojet | 21.75 |
| Silent Service | 21.75 |
| Conflict in Nam | 24.75 |
| Gunship | 21.75 |

BRODERBUND

| | |
|---------------------|-------|
| The Print Shop | 28.75 |
| Graphics Library | 18.75 |
| I, II, III | 19.75 |
| Karateka | 19.75 |
| Bank St. Writer | 32.75 |
| Lode Runner | 21.75 |
| Printshop Companion | 24.75 |
| Bank St. Speller | 32.75 |
| Bank St. Filer | 32.75 |
| Bank St. Mailer | 32.75 |
| Music Shop | 27.75 |

MICROLEAGUE (C-64)

| | |
|-----------|-------|
| Baseball | 24.95 |
| GM disk | 24.95 |
| Team disk | 14.95 |

ACTIVISION (Amiga)

| | |
|---------------|-------|
| Hackler | 26.75 |
| Mind Shadow | 26.75 |
| Music Studio | 29.75 |
| Borrowed Time | 26.75 |

SYNAPSE

| | |
|-------------------|-------|
| Syncalc | 29.95 |
| Template | 14.95 |
| Loderunner Rescue | 19.95 |
| Essex | 24.95 |
| Brimstone | 24.95 |
| Mindwheel | 24.95 |

EPYX-64

| | |
|--------------|-------|
| Fastload | 24.75 |
| Summer Games | 26.75 |

MODEMS

DIGITAL DEVICES

| | |
|-----------------|-------|
| Pocket Modem AT | Call |
| Compuserve | 18.95 |

SUPRA

| | |
|-------------------|-------|
| 1064 Modem (C-64) | 49.95 |
|-------------------|-------|

SUPRA

| | |
|--------------------|--------|
| Supra 300 (Atari) | 49.95 |
| Supra 1200 (Atari) | 199.95 |

US ROBOTICS

| | |
|---------------|-----|
| Password 1200 | 229 |
| Password 300 | 139 |
| Courier 2400 | 469 |

COMMODORE

| | |
|------------|-----|
| 1670 Modem | 155 |
|------------|-----|

MONITORS

ZENITH

| | |
|----------------------|-----|
| ZVM 122A Amber | 75 |
| ZVM 123G Green | 75 |
| ZVM 124 Amber IBM | 129 |
| ZVM 131 Color | 275 |
| ZVM 133 RGB | 369 |
| ZVM 135 Composite | 449 |
| ZVM 136 Hi Res Color | 589 |
| ZVM 1220 | 95 |
| ZVM 1230 | 95 |
| ZVM 1240 | 149 |

PRINCETON GRAPHICS

| | |
|--------------|-----|
| MAX-12 Amber | 165 |
| HX-12 RGB | 465 |
| SR-12 RGB | 595 |

TEKNIKA

| | |
|-----------------|-----|
| MJ-10 Composite | 179 |
| MJ-22 | 255 |

PANASONIC

| | |
|------------------------|-----|
| DTH103 10" RGB Hi Res | 395 |
| TX12H3P 12" Color | 419 |
| TR120MBPA 12" Amber | 109 |
| TR122M9P 12" Green IBM | 148 |
| TR122MYP 12" Amber IBM | 148 |

SAKATA

| | |
|-----------------------|-----|
| SG 1030 12" Green | 99 |
| SA 1000 12" Amber | 109 |
| SG 1500 12" Green TTL | 119 |
| SA 1500 12" Amber TTL | 129 |
| SC 100 13" Color Comp | 209 |
| SC 200 13" RGB | 389 |
| STS1 Tilt Stand | 29 |

COMMODORE

| | |
|------------|------|
| 1902 Color | CALL |
| 1802 Color | CALL |

THOMSON

| | |
|-----------|-----|
| CM36512V1 | 269 |
| CM36632 | 159 |

AMDEK

| | |
|---------------------|-----|
| 300 Green | 118 |
| 300 Amber | 128 |
| 310 Amber IBM | 155 |
| Color 300 Audio | 234 |
| Color 500 Composite | 369 |
| Color 600 | 397 |
| Color 700 | 495 |
| Color 710 | 569 |

NEW HOURS!

Mon-Thur - 9 AM-8 PM
Fri - 9 AM-6 PM
Sat - 10 AM-6 PM

LYCO COMPUTER
America's Mail Order Headquarters

NEW HOURS!

Mon-Thur - 9 AM-8 PM
Fri - 9 AM-6 PM
Sat - 10 AM-6 PM

Lycio Computer Marketing & Consultants



1091 \$228

SAVE ON THESE PRINTERS

COLOR RIBBONS NOW AVAILABLE!!



SG-10 . . . \$205

PANASONIC

| | |
|------------|-----|
| 1091 | 228 |
| 3131 (NEW) | 264 |
| 1092 | 325 |
| 3151 | 409 |
| 1080 (NEW) | 209 |
| 1592 (NEW) | 439 |

OKIDATA

| | |
|------------|-----|
| Okimate 10 | 179 |
| 182 | 214 |
| 192 | 343 |
| 193 | 563 |

EPSON

| | |
|---------------|-----|
| LX80 | 209 |
| FX85 | 333 |
| JX80 | 333 |
| Homewriter 10 | 193 |
| DX10 | 207 |
| DX20 | 297 |
| DX35 | 597 |
| AP-80 | 244 |
| HI-80 | 355 |
| HS-80 | 298 |
| FX-286 (NEW) | 489 |
| LQ-800 (NEW) | 529 |
| LQ-1000 (NEW) | 659 |

LEGEND

| | |
|------|------|
| 1080 | Call |
| 1380 | 258 |
| 1385 | 289 |
| 808 | 148 |

CORONA

| | |
|------------------------|------|
| LP300 | 2495 |
| 200361 Toner Cartridge | 89 |

SILVER REED

| | |
|--------|-----|
| EXP400 | 249 |
| EXP500 | 295 |
| EXP550 | 399 |
| EXP770 | 749 |

JUKI

| | |
|--------------------|-----|
| Juki 6100 | 344 |
| RS232 Serial Board | 55 |
| 6100 Tractor | 119 |
| 6100 Sheet Feeder | 209 |
| Juki 6300 | 757 |

STAR MICRONICS

| | |
|-------------|------|
| SG-10 | 205 |
| SG-10c | 219 |
| SG-15 | 367 |
| SD-10 | 319 |
| SD-15 | 438 |
| SR-10 | 469 |
| SR-15 | 578 |
| SB-10 | 589 |
| PowerType | 297 |
| NX-10 (NEW) | CALL |
| NB-15 (NEW) | CALL |

DIABLO

| | |
|----------|------|
| D25 | 549 |
| 630 API | 1599 |
| 630 ECS | 1759 |
| D 80 1F | 2395 |
| P 32 CQ1 | 699 |
| P J8 | 1749 |
| C 150 | 999 |

TOSHIBA

| | |
|------------------|------|
| P1340 | 469 |
| P351+ | 1149 |
| P341P | 969 |
| P341S | 999 |
| 351 Sheet Feeder | 529 |

CITIZEN

| | |
|------------|-----|
| MSP-10 | 255 |
| MSP-15 | 355 |
| MSP-20 | 337 |
| MSP-25 | 495 |
| 120D | 188 |
| Premier 35 | 429 |

BROTHER

| | |
|-----------|-----|
| HR-15XL-P | 359 |
| HR-15XL-S | 359 |

C. ITOH

| | |
|--------------------|------|
| Prowriter 8510 sp+ | Call |
| 15505 sp+ | Call |
| Printmaster | Call |

SEIKOSHA

| | |
|----------------------|-------|
| SP-1000 VC (C-64) | 185 |
| SP-1000 A Centronics | 199 |
| SP-1000 I IBM | 199 |
| SP-1000 AS RS-232 | 199 |
| SP-1000 AP Apple IIc | 199 |
| BP-5200 I | 649 |
| BP sheet feeder | 850 |
| BP-1000 ribbon | 8.50 |
| BP-5200 ribbon | 12.50 |

DUST COVERS

Atari

| | |
|-------|-------|
| 520ST | 11.95 |
| 130XE | 6.99 |
| 800XL | 6.99 |
| 1050 | 6.99 |
| 1025 | 7.99 |

Commodore

| | |
|-----------|-------|
| C128 | 7.99 |
| 1571/1541 | 6.99 |
| 1902 | 10.95 |
| 1702 | 8.99 |
| C64/Vic20 | 6.99 |

Panasonic

| | |
|-----------|------|
| 1090/1091 | 8.99 |
| 1092 | 8.99 |
| 1093 | 9.99 |

Star Micronics

| | |
|---------|------|
| SG/SD10 | 8.99 |
| SG/SD15 | 9.99 |
| SR10 | 9.99 |
| SR15 | 9.99 |

Okidata

| | |
|-------|------|
| 82/92 | 8.99 |
| 83/93 | 9.99 |
| 193 | 9.99 |

DRIVES

INDUS

| | |
|--------------|-----|
| GT Atari | 195 |
| GT Commodore | 195 |

COMMODORE

| | |
|------|------|
| 1571 | CALL |
| 1541 | CALL |

COMTEL

| | |
|----------------------|-----|
| Enhancer 2000 (C-64) | 159 |
|----------------------|-----|

TANDON

| | |
|--------------------|-----|
| *320K 5 1/4" Drive | 115 |
|--------------------|-----|

INTERFACING

DIGITAL DEVICES

| | |
|------------------------|----|
| Ape Face XLP (Atari) | 49 |
| U-Print A (Atari) | 54 |
| U-Print A 16K Buffer | 74 |
| U-Print AP 16K (Apple) | 99 |

CARDCO

| | |
|--------------|----|
| G-WIZ (C-64) | 54 |
| C/?PS (C-64) | 49 |
| C/?B (C-64) | 39 |

XETEC

| | |
|---------------------|----|
| Super Graphix 64 | 64 |
| Super Graphix JR 64 | 45 |

MICROBITS

| | |
|--------------------|----|
| MPP-1150 (Atari) | 54 |
| MPP-1150XL (Atari) | 59 |
| MicroPrint (Atari) | 39 |

TYMAC

| | |
|-------------------|----|
| Connection (C-64) | 55 |
| Tackler (Apple) | 49 |
| PPC-100 (Apple) | 39 |

MICROTEK

| | |
|----------------------|----|
| Dumpling GX (Apple) | 59 |
| Dumpling 16K (Apple) | 89 |
| RV-611C (Apple) | 49 |

ORANGE MICRO

| | |
|----------------------|-----|
| GRAPPLER+ (Apple) | 85 |
| Grappler 16K (Apple) | 149 |
| ORANGE (Apple) | 59 |
| Grappler CD (C-64) | 79 |

DISKETTES

DENNISON

| | |
|----------------------|-------|
| ELEPHANT 5 1/4" SSDD | 11.99 |
| ELEPHANT 5 1/4" SSDD | 12.99 |
| ELEPHANT 5 1/4" DSDD | 14.99 |
| PREMIUM 5 1/4" SSDD | 13.99 |
| PREMIUM 5 1/4" DSDD | 15.99 |

SUNKYONG

| | |
|-----------------|-------|
| SKC 5 1/4" SSDD | 11.99 |
| SKC 5 1/4" DSDD | 13.99 |

MAXELL

| | |
|------------|-------|
| 5 1/4" MD1 | 13.99 |
|------------|-------|

VERBATIM

| | |
|-------------|-------|
| 5 1/4" SSDD | 13.99 |
| 5 1/4" DSDD | 19.99 |

BONUS

| | |
|-------------|-------|
| 5 1/4" SSDD | 8.99 |
| 5 1/4" DSDD | 12.99 |

3.5" DISKETTES

DENNISON

| | |
|-------------|-------|
| SSDD 5 pak | 14.95 |
| SSDD 10 pak | 26.95 |

MAXELL

| | |
|-------------|-------|
| SSDD 10 pak | 29.95 |
| DSDD 10 pak | 36.95 |

3M

| | |
|-------------|-------|
| SSDD 10 pak | 26.95 |
| DSDD 10 pak | 32.95 |

IBM-PC

MICROPROSE (IBM)

| | |
|--------------------|-------|
| F-15 Strike Eagle | 20.75 |
| Solo Flight | 20.75 |
| Silent Service | 20.75 |
| Decision in Desert | 24.95 |
| Crusade Europe | 24.95 |

SSI (IBM)

| | |
|----------------------|-------|
| Battle for Normandy | 24.95 |
| Knights of Desert | 24.95 |
| Tigers in Snow | 24.95 |
| Computer Baseball | 24.95 |
| Cartels & Cutthroats | 24.95 |
| OP Market Garden | 29.95 |
| 50 Mission Crush | 24.95 |

SUBLOGIC (IBM)

| | |
|------------------|-------|
| Jet Simulator | 34.95 |
| Scenery Disks EA | 14.95 |
| Set 1-6 | 69.95 |

ACTIVISION (IBM)

| | |
|---------------|-------|
| Borrowed Time | 24.75 |
| Mindshadows | 24.75 |
| Music Studio | 29.95 |
| Alter Ego | 29.95 |

SYNAPSE (IBM)

| | |
|--------------------|-------|
| Synstock | 64.95 |
| Essex | 28.95 |
| Wizard of Wall St. | 28.95 |
| Brimstone | 28.95 |

MICROLEAGUE (IBM)

| | |
|--------------|-------|
| M L Baseball | 24.95 |
| General Mgr | 24.95 |
| 85 Team Disk | 14.95 |

LEADING EDGE

| | |
|----------------|--------|
| Nutshell | 69.95 |
| Nutshell Filer | 149.00 |

BRODERBUND (IBM)

| | |
|--------------------|-------|
| Bank St. Writer | 48.95 |
| The Print Shop | 34.95 |
| Graphics Library 1 | 22.95 |
| Ancient Art of War | 27.95 |
| Champ Lode Runner | 22.95 |
| Karateka | 22.95 |

TOLL FREE 1-800-233-8760



TO ORDER



CALL TOLL FREE 1-800-233-8760

In PA 717-494-1030

Customer Service 717-494-1670

or send order to
Lycio Computer
P.O. Box 5088
Jersey Shore, PA
17740

RISK FREE POLICY

In-stock items shipped within 24 hours of order. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the continental U.S. Volume discounts available. PA residents add sales tax. APO, FPO, and international orders add \$5.00 plus 3% for priority mail service. Advertised prices show 4% discount for cash, add 4% for MasterCard or Visa. Personal checks require 4 weeks' clearance before shipping. Ask about UPS Blue and Red label shipping. All merchandise carried under manufacturer's warranty. Free catalog with order. All items subject to change without notice.

your own reinforcement point, and try to block your opponent's armies from this area if you can. If the entry hex is owned but not occupied by your opponent, you'll lose some reinforcements.

After completing a turn, you are credited with additional reinforcements according to how much territory you own. Passing over a hex allows you to claim it; the hex changes color to indicate ownership. Each piece of property provides enough ore and energy to build a new robot, available for use two turns in the future. The numbers in the line of reinforcements are updated after you move to show additional robots being built.

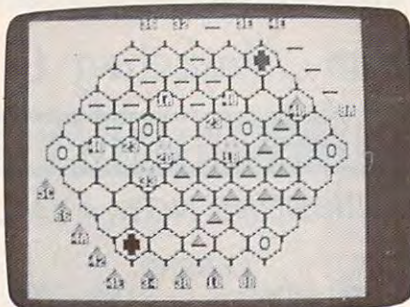
Winning a battle also provides additional armies in the line of reinforcements. As mentioned above, a victorious army captures any dazed enemy bots, which are reprogrammed and available in three turns. At the same time, the winner evacuates injured bots of both sides. Transportation and repair take five turns for friendly bots, seven for enemy bots. The two additional turns are needed for reprogramming the opponent's forces.

If you're losing a battle, the number of injured robots (displayed in the status window) will begin to rise. Remember that, if your opponent reduces your active strength to zero, he or she will capture all of your injured bots; they'll be reprogrammed and added to future reinforcements. To prevent this from happening, you're allowed to bring in a second army for merging. Simply move another army on top of the army with which you want to merge. There's just one rule: One or both of the armies must have a strength less than 32 decimal (1F or less in hex).

Customizing The Scenarios

The five built-in scenarios provide plenty of variety, but if you'd like to add more challenges, here are some suggestions. (The following line number references are for the 128 version of Hex War, Program 1; other versions may differ slightly, although the variable names are the same in most versions.)

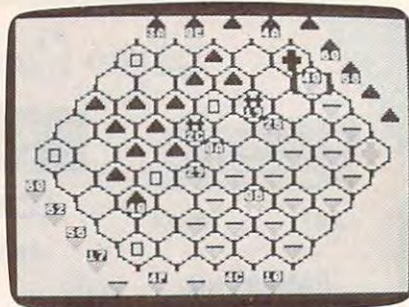
First, a note about the logical organization of the grid. The vari-



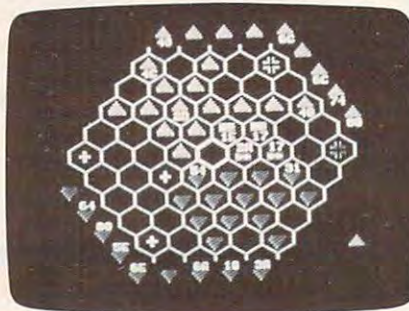
The Commodore 128 version of "Hex War," an absorbing strategy game with many variations.



"Hex War" for the Commodore 64.



"Hex War" For Atari 600XL, 800, 800XL, 1200XL and 130XE



Apple II "Hex War."

ables T and B, CT and CB, and HT and HB are used to locate the coordinates on the playing field (see figure). The first number is T (or HT or CT), the second is B (or HB or CB). These coordinates are also used in the three-dimensional MAP array (where level 0 of the array is the army number, 1 is the current owner and 2 keeps track of whether or not a city is located there); they're also part of the ARMY array. By varying the starting position, number of armies, reinforcement strengths, and location of cities, you could simulate historic battles.

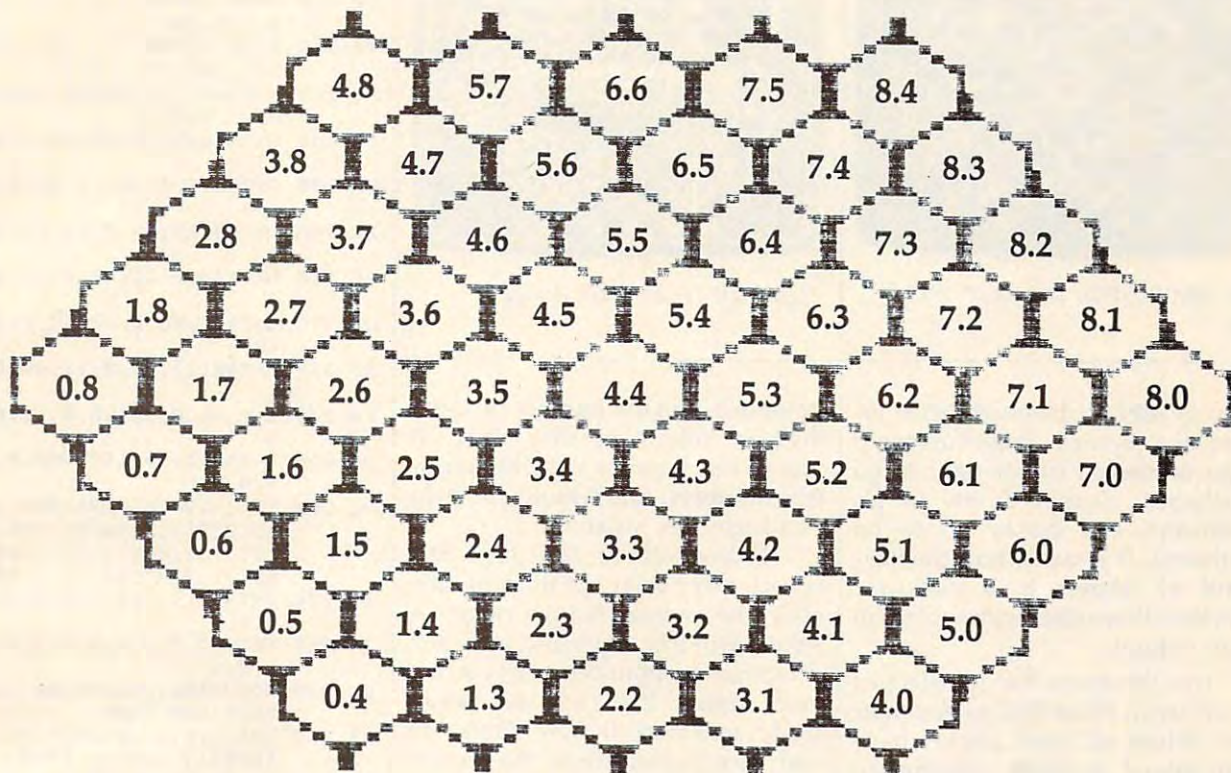
To add or subtract cities from the field, change the value of CN in line 50. You'll also have to change the DATA statements in lines 270 and 280. The numbers there are the T and B coordinates of the cities.

The strengths and locations of the armies can be changed as well. The DATA statements starting at line 1540 determine the strength (64) and T/B coordinates for the armies at the beginning of the game. If you wish to start with more armies (or fewer), you'll have to change the inner FOR-NEXT loop

(with the index of K) in line 1500. In that same line, change NX(J) to one number higher than the number of armies on each side. For example, if you want six armies apiece, change NX(J) to 7. The subroutine at line 1600 sets up the reinforcements; if you don't like the random patterns, change the formula here.

Variables defined in lines 70-90 control the play of the game. PN determines which player goes second; it can be either zero or one. Variable ME controls the maximum merge strength. If you'd like to be able to merge any two armies, change it to a high value (512, for example). To remove the merge option altogether, change ME to zero.

The movement points are defined by MM in line 80. Movement across friendly territory takes one point, across neutral or hostile territory two points. Increasing MM will give your armies more mobility. The three variables KA, KB, and KC affect the outcome of individual battles. KA determines how many bots are vaporized, KB controls the number injured, and KC affects how many are dazed. If you make the fractions smaller (1/24, for



example), the battles end more quickly. The subroutine starting at line 2600 resolves current battles.

Commodore 64 Version

The 64 version of Hex War (Program 2) looks and plays exactly like the original 128 version. However, one additional step is needed before you run the game. After you have typed in the game and saved it on disk or tape, type this line in direct mode (without line numbers):

`POKE 44,64:POKE 64*256,0:NEW`

Be sure to press RETURN after you type the line. Now load and run the program as usual. It is very important that you perform this step before running the program: If you don't, the screen will be jumbled and impossible to decipher.

You may find it easier to let the 64 handle this chore for you. Program 3 is a short loader which performs the setup, then loads and runs Hex War. To use the loader, you must have Program 2 saved with the name HEX WAR on the same disk or tape as Program 3 (for tape, Program 2 must follow Pro-

gram 3 on the tape and the DV=8 in line 10 of Program 3 must be changed to DV=1).

Atari Version

In the Atari version (Program 4), armies are maneuvered using a joystick plugged into port 1. Joystick controls are the same as described above for the 128 version.

This version generates extra colors in graphics mode 0 using a technique known as artifacting. However, the resulting colors may vary on different Ataris, so a small change may be required. The game should start with the red army at the top of the screen and the blue army at the bottom. If that is not the case on your machine, change the following line:

```
ND 4077 RESTORE 4140:FOR A=C
HSET+240 TO CHSET+24
7:READ B:POKE A,B:PO
KE A+8,B*2:NEXT A:RE
TURN
```

If the colors are not corrected, the machine may appear to declare the wrong side the winner at the end of the game.

Apple Version

Program 5 is for all Apple II-series computers using either DOS 3.3 or ProDOS. To get the full benefit of the detailed high-resolution graphics, a color monitor or color TV is recommended. (The program incorporates the HROUT high-resolution graphics routine from the "Apple Superfont" article in the April 1985 issue to generate these graphics.)

Use keyboard controls to maneuver armies in this version. Press the period (.) key to go northeast (use the > symbol on that key as a reminder), the comma (,) key to go northwest (note the < symbol on that key), the → to move southeast, and the ← key to move southwest. Use the space bar to select or set an army. Press RETURN to end your turn before all your armies have been moved. A player indicator appears in the lower-right corner to indicate which player has the current turn.

IBM PC/PCjr Version

Like the Apple version, the IBM PC/PCjr version of Hex War (Pro-


```

{ 2 SPACES }Q";NEXTK:PRI
NT:NEXTJ
QX 630 FORJ=1TO5:PRINTSPC(J*2)
;:FORK=1TO10-J:PRINT"R
{ 2 SPACES }E";NEXTK:PRI
NT
SS 640 PRINTSPC(J*2+1);:FORK=1
TO9-J:PRINT"QW
{ 2 SPACES }";NEXTK:PRI
T"QW"
PX 650 NEXTJ
CD 660 C$="UI{DOWN}{ 2 LEFT}JK"
:D$="[RVS]{V}{C}{DOWN}
{ 2 LEFT}{F}{D}"
MJ 670 COLOR5,3:FORJ=1TO12:GOS
UB710:NEXT
HQ 680 J=1:COLOR5,7:IFGN=1THEN
C$=D$:GOSUB710:J=2:COLO
R5,5:GOSUB710
RB 690 IFGN=2THENC$=D$:GOSUB71
0:J=3:COLOR5,5:GOSUB710
SH 700 PRINT"HOME";:RETURN
KB 710 K=CIT(J,0):L=CIT(J,1):X
=(K-L)*2+19:Y=(12-(K+L)
)*2+3:CHAR1,X,Y,C$:RETU
RN
FF 800 IFNX(PN)<2THENRETURN
AX 805 HT=4:HB=4:GOSUB1000:SPR
ITEL,1,1,0
KE 810 MV=0:CT=0:CB=0:PK=0:REM
PICKED UP OR NOT
HS 820 K=0:FORJ=1TONX(PN)-1:IF
(ARMY(J,0,PN)>0)AND(ARM
Y(J,6,PN)<1)THENK=1:J=N
X(PN)-1
KX 830 NEXTJ:IFK=0 THEN RETURN
SB 840 GETG$:IFG$=CHR$(13)THEN
RETURN
HD 850 J=JOY(2):IFJ=0THEN840:E
LSE IF(JAND128)THEN1100
:ELSE IF(JAND1)=0THEN84
0
KK 860 J=(J-1)/2:IFJAND1THENB1
=HB+J-2:T1=HT:ELSET1=HT
+1-J:B1=HB
HD 870 IF (T1<0)OR(T1>8)THEN84
0
FP 880 IF (B1<0)OR(B1>8)THEN84
0
KC 890 S1=T1+B1:IF(S1<4)OR(S1>
12)THEN840
EM 900 HB=B1:HT=T1:GOSUB1000:W
INDOW1,1,8,4,1:QN=MAP(H
T,HB,0):IFQN=0THENPRINT
"{ 2 HOME}";:GOTO840:ELS
E Q1=MAP(HT,HB,1)-1
JF 910 COLOR5,7-2*Q1:PRINTUSIN
G"[RVS]###";QN;:PRINT"***
****[OFF]{5}"
RH 920 FORJ=0TO3:PRINTUSIN"##
##";:ARMY(QN,J,Q1);:NEXT
DD 930 PRINT"{ 2 HOME}";:GOTO84
0
PA 1000 SX=172+16*(HT-HB):SY=2
64-16*(HT+HB)
KC 1010 MOVSPR1,SX,SY
SA 1020 IF MAP(HT,HB,2)=1THENS
PRITEL,1,3,0:RETURN
KP 1030 SPRITEL,1,1,0:RETURN
CX 1100 IFJOY(2)<>0THEN1100
FE 1110 IFPK=1THEN1200:REM PIC
KEDUP, CHECK IF OK
FD 1120 IF((MAP(HT,HB,1)<>PN+1)
)OR(MAP(HT,HB,0)=0)TH
ENGOTO810:REM NO ONE H
OME
XE 1130 AN=MAP(HT,HB,0):IFARMY
(AN,6,PN)<>0THEN810:RE
M ARMY AN IS ENGAGED
SD 1140 PK=1:CT=HT:CB=HB:CS=AR
MY(AN,0,PN):REM T&B, C
URRENT STRENGTH
DP 1150 SOUND1,5000,10
MK 1160 GOTO840
XP 1200 J=((HT=CT)AND(HB=CB))
AQ 1210 IFJAND(MV=0)THEN810
RE 1220 IFJAND(MV>0)THEN1420
MH 1230 AS=ARMY(MAP(HT,HB,0),0
,PN):IF((AS>ME)AND(CS>
ME))OR((MAP(HT,HB,1)-1
=1-PN)AND(AS>0)) THEN8
40
FB 1240 DT=ABS(CT-HT):DB=ABS(C
B-HB):TL=DT+DB:IF NOT(
(TL=1)OR((CT+CB=HT+HB)
AND(DT=1)))THEN840
FS 1250 MG=MAP(HT,HB,0): IF MG
=0THEN1300
AE 1260 FORJ=0TO3:ARMY(MG,J,PN
)=ARMY(MG,J,PN)+ARMY(A
N,J,PN):ARMY(AN,J,PN)=
0:NEXTJ
SM 1270 ARMY(MG,6,PN)=1:MAP(CT
,CB,0)=0
AD 1280 CS=ARMY(MG,0,PN):AN=MG
:MV=MM+1
RH 1290 GOTO 1380
FF 1300 NB=MAP(HT,HB,1)-1:MV=M
V+1:IF(NB<>PN)THENMV=M
V+1
DK 1310 MAP(CT,CB,0)=0
SA 1320 MAP(HT,HB,0)=AN:MAP(HT
,HB,1)=PN+1:ARMY(AN,4
,PN)=HT:ARMY(AN,5,PN)=H
B:IF MV>=MMTHEN ARMY(A
N,6,PN)=1
KR 1330 K=0:FORJ=-1TO1STEP2:J1
=HT+J:J2=HB+J:J3=HB-J:
IF(J1<0)OR(J1>8)THEN13
40:ELSE IF(MAP(J1,HB,0)
)>0)THENIF(MAP(J1,HB,1)
)=2-PN)THENK=1:J=1:GOT
O1360
BR 1340 IF(J2<0)OR(J2>8)THEN13
50:ELSE IF(MAP(HT,J2,0)
)>0)THENIF(MAP(HT,J2,1)
)=2-PN)THENK=1:J=1:GOT
O1360
KE 1350 IF(J3<0)OR(J3>8)OR(J1<
0)OR(J1>8)THEN1360:ELS
E IF(MAP(J1,J3,0)>0)TH
ENIF(MAP(J1,J3,1)=2-PN)
THENK=1:J=1
HK 1360 NEXTJ: REM ZOC
AS 1370 IFK=1THEN ARMY(AN,6,PN
)=1:MV=MM+1
EG 1380 A=PN:J=CT:K=CB:C=0:D=0
:GOSUB1830
QP 1390 J=HT:K=HB:C=CS:D=ARMY(
AN,6,PN):GOSUB1830
XJ 1400 CT=HT:CB=HB
MQ 1410 IFMV<MMTHEN840
HA 1420 ARMY(AN,6,PN)=1:J=HT:K
=HB:C=CS:D=1:GOSUB1830
SG 1430 GOTO810
FC 1500 RESTORE1540:FORJ=0TO1:
NX(J)=5:FORK=1TO4:READ
A,B,C
QM 1510 ARMY(K,0,J)=A:ARMY(K,4
,J)=B:ARMY(K,5,J)=C:MA
P(B,C,0)=K:MAP(B,C,1)=
J+1
BJ 1520 NEXTK,J
EK 1530 REM STRENGTH, T-POS, B
-POS
JD 1540 DATA 64,2,8,64,3,7,64,
5,6,64,6,6:REM BLUE
QP 1550 DATA 64,2,2,64,3,2,64,
5,1,64,6,0:REM VIOLET
JK 1600 REM SET RANDOM REINFOR
CEMENTS
AS 1610 FORJ=0TO1:FORK=0TO20
QM 1620 A=INT(RND(1)*K*3):FORL
=1TO5:A=A+INT(RND(1)*2
1-8):NEXTL:IFA<16THENA
=0:ELSEA=(A+K*8)AND254
HA 1630 FQ(K,J)=A:NEXTK,J
RG 1640 RETURN
EH 1700 REM ARMIES->MAP
SC 1710 FORJ=0TO8:FORK=0TO8
FX 1720 A=MAP(J,K,1):IFATHENA=
A-1:GOSUB1800
HH 1730 NEXTK,J
PH 1740 FORA=0TO1:E=13+A*12:F=
A*22:DX=2-4*A:D=0
PF 1750 FORJ=0TO8:C=FQ(J,A):GO
SUB1840
HF 1760 E=E+DX*2:IFJ>3THENF=F+
DX:E=E-DX
XC 1770 NEXTJ,A
CA 1780 RETURN
QC 1800 B=MAP(J,K,0)
PH 1810 C=ARMY(B,0,A)
BK 1820 D=ARMY(B,6,A)
FJ 1830 E=(J-K+10)*2-1:F=(13-J
-K)*2+1:REM T&B TO X/
Y
SC 1840 CHAR1,E,F:IFATHENPRINT
P1$:ELSEPRINTP0$
MH 1850 IFC=0THENRETURN
PP 1860 COLOR5,(7-2*A):CHAR1,E
,F+A
HR 1870 PRINTCHR$(18);RIGHT$(H
EX$(C),2);CHR$(146)
RD 1880 IFDTHENF=F+1-A:G=1024+
E+F*40:POKEG,43:POKEG+
1,43:REM ++
XK 1890 RETURN
GC 1900 SW=0:E=NX(PN)-1:IFE<1T
HENRETURN
XJ 1910 FORJ=1TOE-1:IFARMY(J,0
,PN)<1THENBEGIN
SE 1920 T=ARMY(J,4,PN):B=ARMY(
J,5,PN):IFMAP(T,B,0)=J
THENMAP(T,B,0)=0
BP 1930 FORK=JTOE:FORL=0TO6:AR
MY(K,L,PN)=ARMY(K+1,L
,PN):ARMY(K+1,L,PN)=0:N
EXTL
JQ 1940 T=ARMY(K,4,PN):B=ARMY(
K,5,PN):MAP(T,B,0)=K
PH 1950 NEXTK
MA 1960 NX(PN)=NX(PN)-1:J=E:SW
=1:BEND
FJ 1970 NEXTJ:IF SW THEN1900
MG 2000 FORJ=1TOE:ARMY(J,0,PN)
=ARMY(J,0,PN)+ARMY(J,2
,PN)
SD 2010 ARMY(J,2,PN)=ARMY(J,3
,PN):ARMY(J,3,PN)=0
QS 2020 ARMY(J,6,PN)=0
PP 2030 NEXTJ:K=NX(1-PN):FOR J
=1TOK:ARMY(J,6,1-PN)=0
:NEXT
XF 2040 GOSUB2400
DS 2050 IFBP>0 THEN FORJ=0TO1:
FORK=1TOBP:A=BTL(K,J,0
):ARMY(A,6,J)=ARMY(A,6
,J)+1:NEXTK,J
CA 2060 RETURN
KG 2100 GOSUB2400
BX 2110 A=NX(0):IFNX(1)>ATHENA
=NX(1)
EX 2120 FORJ=0TO1:FORK=1TOA:AR
MY(K,6,J)=0:NEXTK,J
GP 2130 GOSUB2050
SH 2140 RETURN
PB 2200 FORJ=0TO1:A=1-J:B=NX(J
)-1
DD 2210 FORK=1TOB
EM 2220 IF ARMY(K,0,J)<1 THEN
{SPACE}BEGIN
JC 2230 FQ(2,A)=FQ(2,A)+ARMY(K

```



```

,2,J)+ARMY(K,3,J):IF F
Q(2,A)>255 THEN C=FQ(2
,A)-255:FQ(3,A)=FQ(3,A
)+C:FQ(2,A)=255
JX 2240 FQ(6,A)=FQ(6,A)+ARMY(K
,1,J):IF FQ(6,A)>255 T
HEN C=FQ(6,A)-255:FQ(7
,A)=FQ(7,A)+C:FQ(6,A)=
255
JQ 2250 IF (MAP(ARMY(K,4,J),AR
MY(K,5,J),0)=K)AND(MAP
(ARMY(K,4,J),ARMY(K,5
,J),1)=J+1) THEN MAP(AR
MY(K,4,J),ARMY(K,5,J),
0)=0
RB 2260 FORL=0TO6:ARMY(K,L,J)=
0:NEXTL
BB 2270 BEND
FR 2280 IF ARMY(K,6,J)<1 THEN
{SPACE}BEGIN:REM EVACU
ATE INJURED
AE 2290 FQ(4,J)=FQ(4,J)+ARMY(K
,1,J): ARMY(K,1,J)=0
DJ 2300 IF FQ(4,J)>255 THEN C=
FQ(4,J)-255:FQ(5,J)=FQ
(5,J)+C:FQ(4,J)=255
KA 2310 BEND
XP 2320 NEXTK,J:RETURN
SJ 2400 BP=0
KG 2410 FORJ=0TO8:J1=(J-4)*(4-
J)>0:J2=8-(J+4)*(4-J):
FORK=J1TOJ2
JE 2420 A=MAP(J,K,0)
HQ 2430 R=MAP(J,K,1)
BG 2440 IF (A=0)OR(R=0) THEN24
90
KQ 2450 IF ARMY(A,0,R-1)<1 THE
N2490
DK 2460 T=J+1:B=K:GOSUB2500
HK 2470 B=B-1:GOSUB2500
BP 2480 T=T-1:GOSUB2500
RH 2490 NEXTK,J:RETURN
RR 2500 IF (T<0)OR(B<0)OR(T>8)O
R(B>8)THEN RETURN
MA 2510 PA=MAP(T,B,0):IF PA=0
{SPACE}THEN RETURN
FG 2520 IF MAP(T,B,1)=R THEN R
ETURN
FX 2530 IF ARMY(PA,0,2-R)<1 TH
EN RETURN
MA 2540 BP=BP+1:BTL(BP,R-1,0)=
A:BTL(BP,2-R,0)=PA:RET
URN
CF 2600 IFBP=0THENRETURN
FC 2610 FORJ=1TOBP
XM 2620 FORK=0TO1:A=1-K
QX 2630 AN=BTL(J,K,0)
BF 2640 AS=ARMY(AN,0,K):HT=ARM
Y(AN,6,K):CT=INT(AS/HT
)+1
GH 2650 BTL(J,A,1)=INT(CT*K+1
)
JC 2660 BTL(J,A,2)=INT(CT*K+1
)
MB 2670 BTL(J,A,3)=INT(CT*K+1
)
XC 2680 NEXTK,J
MD 2700 FORJ=1TOBP:J0=BTL(J,0,
0):J1=BTL(J,1,0)
BR 2710 GOSUB3100
CG 2720 ARMY(J0,0,0)=ARMY(J0,0
,0)-A*BTL(J,0,1)
SK 2730 ARMY(J1,0,1)=ARMY(J1,0
,1)-B*BTL(J,1,1)
MS 2740 GOSUB3100
BC 2750 C=A*BTL(J,0,2):ARMY(J0
,0,0)=ARMY(J0,0,0)-C:A
RMY(J0,1,0)=ARMY(J0,1,
0)+C
FJ 2760 C=B*BTL(J,1,2):ARMY(J1
,0,1)=ARMY(J1,0,1)-C:A
RMY(J1,1,1)=ARMY(J1,1,
1)+C
QD 2770 GOSUB3100
AX 2780 C=A*BTL(J,0,3):ARMY(J0
,0,0)=ARMY(J0,0,0)-C:A
RMY(J0,3,0)=ARMY(J0,3,
0)+C
FF 2790 C=B*BTL(J,1,3):ARMY(J1
,0,1)=ARMY(J1,0,1)-C:A
RMY(J1,3,1)=ARMY(J1,3,
1)+C
XP 2800 NEXTJ
CB 2810 RETURN
GQ 2900 A=1-PN:B=0
GG 2910 FORJ=0TO8:FORK=0TO8
PA 2920 IFMAP(J,K,1)=PN+1THENB
=B+1
KD 2930 NEXTK,J
JB 2950 FQ(1,PN)=FQ(1,PN)+B: I
F FQ(1,PN)>255 THEN B=
FQ(1,PN)-255:FQ(2,PN)=
FQ(2,PN)+B:FQ(1,PN)=25
5
GQ 2960 T=4:B=PN*8
SJ 2970 IF MAP(T,B,0)<>0 THEN
{SPACE}RETURN
JK 2980 IF MAP(T,B,1)=PN+1 THE
N FQ(0,A)=0:FQ(1,A)=0:
GOTO3060
XS 2990 J=NK(A):IFJ>31THENRETU
RN
HR 3000 J1=FQ(0,A):IF J1<1 THE
N3060
EP 3010 NK(A)=NK(A)+1
XE 3020 MAP(T,B,0)=J:MAP(T,B,1
)=A+1
BK 3030 ARMY(J,0,A)=J1
QR 3040 FORK=1TO3:ARMY(J,K,A)=
0:NEXTK
PM 3050 ARMY(J,4,A)=T:ARMY(J,5
,A)=B
KH 3060 FORK=0TO19:FQ(K,A)=FQ(
K+1,A):NEXTK
PK 3070 FQ(20,A)=0
BA 3080 RETURN
FS 3100 A=0:FORM=1TO6:IFRND(1)
<.5THENA=A+1
AH 3110 NEXTM:B=6-A:RETURN
PS 3200 WINDOW6,7,33,16,1
EJ 3210 PRINT"[7][2 M][2 L]
{RVS}[2 K][J][H][G] SC
ENARIO [M][N][L]{OFF}
[3 K][J][2 G][M]"
FB 3220 PRINT"[LEFT][G][M]
{2 SPACES}1> CAPTURE C
APITAL/FAR{2 SPACES}
[G]"
QX 3230 PRINT"[M]{2 SPACES}2>
{SPACE}CAPTURE CAPITAL
/NEAR [G]"
GQ 3240 PRINT"[M]{2 SPACES}3>
{SPACE}OCCUPY
{3 SPACES}8/12 CITIES
{SPACE}[G]"
AR 3250 PRINT"[M]{2 SPACES}4>
{SPACE}CONTROL
{2 SPACES}6/12 CITIES
{SPACE}[G]"
DJ 3260 PRINT"[M]{2 SPACES}5>
{SPACE}OCCUPY
{2 SPACES}40/61 HEXES
{2 SPACES}[G][M]";:PRI
NTSPC(26);"[G]"
SQ 3270 PRINT"[2 M][2 L]{RVS}
[2 K][J][H][G]
{10 SPACES}[M][N][L]
{OFF}[3 K][J][2 G]"
GE 3280 GETKEYAS:GN=VAL(AS):IF
GN<10RND>5THEN3280
DP 3290 CHAR1,2,1+GN,"ZZ":SLEE
P1:PRINT"[2 HOME]"
FX 3300 RETURN
GS 3400 A=0:ON GN GOSUB 3430,3
450,3480,3490,3580
JD 3410 IFA=0THENRETURN:ELSEEN
S=C$=Q$=A:GOSUB600:GOS
UB1710:A=Q$
KD 3420 PRINT"[HOME]PLAYER";A;
" WINS":PRINTEN$:PRINT
"(PRESS ANY KEY)":POKE
208,0:GETKEYAS:RUN
BB 3430 IF MAP(CIT(2,0),CIT(2,
1),1)=1THENA=2:C$="BLU
E CAPTURED THE CAPITAL
":RETURN
CR 3440 GOTO3460
SG 3450 IF MAP(CIT(3,0),CIT(3,
1),1)=1THENA=2:C$="BLU
E CAPTURED THE CAPITAL
":RETURN
KB 3460 IF MAP(CIT(1,0),CIT(1,
1),1)=2THENA=1:C$="VIO
LET CAPTURED THE CAPIT
AL"
FP 3470 RETURN
BC 3480 L=8:GOTO3500
JB 3490 L=6
FG 3500 C(1)=0:C(2)=0
PK 3510 FORJ=1TO12:T=CIT(J,0):
B=CIT(J,1)
MB 3520 R=MAP(T,B,1):C(R)=C(R)
+1
DB 3530 IF GN=4THEN AN=MAP(T,B
,0):IF R>0 THEN IF (AN
=0)OR(ARMY(AN,6,R-1)>0
)THEN C(R)=C(R)-1
KJ 3540 NEXTJ
CB 3550 IF C(1)=> L THEN A=2:C
$="BLUE HAS CAPTURED"+
STR$(C(1))+ " CITIES":R
ETURN
QF 3560 IF C(2)=> L THEN A=1:C
$="VIOLET HAS CAPTURED
"+STR$(C(2))+ " CITIES"
AB 3570 RETURN
RK 3580 C(1)=0:C(2)=0
PP 3590 FORJ=0TO8:FORK=0TO8
KM 3600 R=MAP(J,K,1):C(R)=C(R)
+1
SR 3610 NEXTK,J
CE 3620 IF C(1)=>40THENA=2:C$=
"BLUE OCCUPIES"+STR$(C
(1))+ " HEXES":RETURN
RK 3630 IF C(2)=>40THENA=1:C$=
"VIOLET OCCUPIES"+STR$
(C(2))+ " HEXES"
DF 3640 RETURN

```

Program 2. Hex War For Commodore 64

Version by Kevin Mykytyn, Editorial Programmer

```

EB 0 POKE53269,0:PRINT"[CLR]":
XC=14:YC=12:GOSUB1:PRINT
PLEASE WAIT":GOTO 9
ME 1 POKE781,YC:POKE782,XC:POK
E783,0:SYS65520:RETURN
GB 2 QV=15-(PEEK(56320)AND15):
J=JY(QV)-128*((PEEK(56320
)AND16)=0):RETURN
JQ 3 QV=QVAND255:H$="012345678
9ABCDEF":Z$=MID$(H$,INT(Q
V/16)+1,1)
EH 4 Z$=Z$+MID$(H$,QV-INT(QV/1
6)*16+1,1):RETURN
XC 9 IFPEEK(12289)+PEEK(12290)
=212THEN20

```



```

RP 1352 IF (MAP(J1,J3,0)>0) THEN
  IF (MAP(J1,J3,1)=2-PN) T
  HENK=1:J=1
RB 1360 NEXTJ
AS 1370 IFK=1 THEN ARMY(AN,6,PN)
  =1:MV=MM+1
EG 1380 A=PN:J=CT:K=CB:C=0:D=0
  :GOSUB1830
QP 1390 J=HT:K=HB:C=CS:D=ARMY(
  AN,6,PN):GOSUB1830
XJ 1400 CT=HT:CB=HB
MQ 1410 IFMV<MM THEN 840
HA 1420 ARMY(AN,6,PN)=1:J=HT:K
  =HB:C=CS:D=1:GOSUB1830
SG 17430 GOTO810
GR 1500 RESTORE:FORZ=1 TO 135:RE
  ADB:NEXT:FORJ=0 TO 1:NX(
  J)=5:FORK=1 TO 4:READA,B
  ,C
QM 1510 ARMY(K,0,J)=A:ARMY(K,4
  ,J)=B:ARMY(K,5,J)=C:MA
  P(B,C,0)=K:MAP(B,C,1)=
  J+1
B7J 1520 NEXTK,J
EK 1530 REM STRENGTH, T-POS, B
  -POS
JD 1540 DATA 64,2,8,64,3,7,64,
  5,6,64,6,6:REM BLUE
QP 1550 DATA 64,2,2,64,3,2,64,
  5,1,64,6,0:REM VIOLET
JK 1600 REM SET RANDOM REINFOR
  CEMENTS
PB 1610 FORJ=0 TO 1:FORK=0 TO 20:A
  =INT(RND(1)*K*3):FORL=
  1 TO 5:A=A+INT(RND(1)*21
  -8)
JR 1620 NEXTL:IFA<16 THEN A=0:GO
  TO 1630
XF 1625 A=(A+K*8) AND 254
HA 1630 FQ(K,J)=A:NEXTK,J
RG 1640 RETURN
EH 1700 REM ARMIES->MAP
SC 1710 FORJ=0 TO 8:FORK=0 TO 8
FX 1720 A=MAP(J,K,1):IFATHENA=
  A-1:GOSUB1800
HH 1730 NEXTK,J
PH 1740 FORA=0 TO 1:E=13+A*12:F=
  A*22:DX=2-4*A:D=0
PF 1750 FORJ=0 TO 8:C=FQ(J,A):GO
  SUB1840
HF 1760 E=E+DX*2:IFJ>3 THEN F=F+
  DX:E=E-DX
XC 1770 NEXTJ,A
CA 1780 RETURN
QC 1800 B=MAP(J,K,0)
PH 1810 C=ARMY(B,0,A)
BK 1820 D=ARMY(B,6,A)
FJ 1830 E=(J-K+10)*2-1:F=(13-J
  -K)*2+1:REM T&B TO X/
  Y
SK 1840 XC=E:YC=F:GOSUB1:IFATH
  ENPRINTP1$:GOTO1850
SB 1845 PRINTP0$
MH 1850 IFC=0 THEN RETURN
DF 1860 POKEC5,6-2*A:XC=E:YC=F
  +A:GOSUB1
CF 1870 QV=C:GOSUB3:PRINTCHR$(
  18):Z$:CHR$(146)
RD 1880 IFD THEN F=F+1-A:G=1024+
  E+F*40:POKEG,43:POKEG+
  1,43:REM ++
XK 1890 RETURN
GC 1900 SW=0:E=NX(PN)-1:IFE<1 T
  HEN RETURN
SP 1910 FORJ=1 TO E-1:IFARMY(J,0
  ,PN)=1 THEN 1970
SE 1920 T=ARMY(J,4,PN):B=ARMY(
  J,5,PN):IFMAP(T,B,0)=J
  THEN MAP(T,B,0)=0
BP 1930 FORK=J TO E:FORL=0 TO 6:AR
  MY(K,L,PN)=ARMY(K+1,L,
  PN):ARMY(K+1,L,PN)=0:N
  EXT
JQ 1940 T=ARMY(K,4,PN):B=ARMY(
  K,5,PN):MAP(T,B,0)=K
PH 1950 NEXTK
CA 1960 NX(PN)=NX(PN)-1:J=E:SW
  =1
FJ 1970 NEXTJ:IF SW THEN 1900?
MG 2000 FORJ=1 TO E:ARMY(J,0,PN)
  =ARMY(J,0,PN)+ARMY(J,2
  ,PN)
SD 2010 ARMY(J,2,PN)=ARMY(J,3,
  PN):ARMY(J,3,PN)=0
QS 2020 ARMY(J,6,PN)=0
PP 2030 NEXTJ:K=NX(1-PN):FOR J
  =1 TO K:ARMY(J,6,1-PN)=0
  :NEXT
XF 2040 GOSUB2400
MC 2050 IFBP>0 THEN FORJ=0 TO 1:FO
  RK=1 TO BP:A=BTL(K,J,0):
  AR(A,6,J)=AR(A,6,J)+1:
  NEXTK,J
CA 2060 RETURN
KG 2100 GOSUB2400
BX 2110 A=NX(0):IFNX(1)>ATHENA
  =NX(1)
EX 2120 FORJ=0 TO 1:FORK=1 TO A:AR
  MY(K,6,J)=0:NEXTK,J
GP 2130 GOSUB2050
SH 2140 RETURN
PB 2200 FORJ=0 TO 1:A=1-J:B=NX(J
  )-1
DD 2210 FORK=1 TO B
JQ 2220 IF ARMY(K,0,J)>1 THEN
  2280
QD 2230 FQ(2,A)=FQ(2,A)+ARMY(K
  ,2,J)+ARMY(K,3,J)
XQ 2235 IF FQ(2,A)>255 THEN C=
  FQ(2,A)-255:FQ(3,A)=FQ
  (3,A)+C:FQ(2,A)=255
RM 2240 FQ(6,A)=FQ(6,A)+ARMY(K
  ,1,J)
JC 2245 IF FQ(6,A)>255 THEN C=
  FQ(6,A)-255:FQ(7,A)=FQ
  (7,A)+C:FQ(6,A)=255
XK 2250 IF MAP(ARMY(K,4,J),ARM
  Y(K,5,J),0)<>K THEN 22
  60
DD 2252 IF MAP(ARMY(K,4,J),ARM
  Y(K,5,J),1)<>J+1 THEN
  {SPACE}2260
RR 2253 MAP(ARMY(K,4,J),ARMY(K
  ,5,J),0)=0
RB 2260 FORL=0 TO 6:ARMY(K,L,J)=
  0:NEXTL
AS 2280 IF ARMY(K,6,J)>1 THEN
  2320:REM EVACUATE INJ
  URED
AE 2290 FQ(4,J)=FQ(4,J)+ARMY(K
  ,1,J):ARMY(K,1,J)=0
DJ 2300 IF FQ(4,J)>255 THEN C=
  FQ(4,J)-255:FQ(5,J)=FQ
  (5,J)+C:FQ(4,J)=255
XP 2320 NEXTK,J:RETURN
SJ 2400 BP=0
KG 2410 FORJ=0 TO 8:J1=(J-4)*(4-
  J+0):J2=8-(J+4)*(4-J):
  FORK=J1 TO J2
JE 2420 A=MAP(J,K,0)
HQ 2430 R=MAP(J,K,1)
BG 2440 IF (A=0) OR (R=0) THEN 24
  90
KQ 2450 IF ARMY(A,0,R-1)<1 THE
  N 2490
DK 2460 T=J+1:B=K:GOSUB2500
HK 2470 B=B-1:GOSUB2500
BP 2480 T=T-1:GOSUB2500
RH 2490 NEXTK,J:RETURN
RR 2500 IF (T<0) OR (B<0) OR (T>8) O
  R (B>8) THEN RETURN
MA 2510 PA=MAP(T,B,0):IF PA=0
  {SPACE} THEN RETURN
FG 2520 IF MAP(T,B,1)=R THEN R
  ETURN
FX 2530 IF ARMY(PA,0,2-R)<1 TH
  EN RETURN
MA 2540 BP=BP+1:BTL(BP,R-1,0)=
  A:BTL(BP,2-R,0)=PA:RET
  URN
CF 2600 IFBP=0 THEN RETURN
FC 2610 FORJ=1 TO BP
XM 2620 FORK=0 TO 1:A=1-K
QX 2630 AN=BTL(J,K,0)
BF 2640 AS=ARMY(AN,0,K):HT=ARM
  Y(AN,6,K):CT=INT(AS/HT
  )+1
GH 2650 BTL(J,A,1)=INT(CT*K A+1
  )
JC 2660 BTL(J,A,2)=INT(CT*K B+1
  )
MB 2670 BTL(J,A,3)=INT(CT*K C+1
  )
XC 2680 NEXTK,J
MD 2700 FORJ=1 TO BP:J0=BTL(J,0,
  0):J1=BTL(J,1,0)
BR 2710 GOSUB3100
CG 2720 ARMY(J0,0,0)=ARMY(J0,0,
  0)-A*BTL(J,0,1)
SK 2730 ARMY(J1,0,1)=ARMY(J1,0,
  1)-B*BTL(J,1,1)
MS 2740 GOSUB3100
BC 2750 C=A*BTL(J,0,2):ARMY(J0,
  0,0)=ARMY(J0,0,0)-C:A
  RMY(J0,1,0)=ARMY(J0,1,
  0)+C
FJ 2760 C=B*BTL(J,1,2):ARMY(J1,
  0,1)=ARMY(J1,0,1)-C:A
  RMY(J1,1,1)=ARMY(J1,1,
  1)+C
QD 2770 GOSUB3100
AX 2780 C=A*BTL(J,0,3):ARMY(J0,
  0,0)=ARMY(J0,0,0)-C:A
  RMY(J0,3,0)=ARMY(J0,3,
  0)+C
FF 2790 C=B*BTL(J,1,3):ARMY(J1,
  0,1)=ARMY(J1,0,1)-C:A
  RMY(J1,3,1)=ARMY(J1,3,
  1)+C
XP 2800 NEXTJ
CB 2810 RETURN
GQ 2900 A=1-PN:B=0
GG 2910 FORJ=0 TO 8:FORK=0 TO 8
PA 2920 IFMAP(J,K,1)=PN+1 THEN B
  =B+1
KD 2930 NEXTK,J
DM 2950 FQ(1,PN)=FQ(1,PN)+B
HG 2955 IF FQ(1,PN)>255 THEN B=FQ
  (1,PN)-255:FQ(2,PN)=FQ
  (2,PN)+B:FQ(1,PN)=255
GQ 2960 T=4:B=PN*8
SJ 2970 IF MAP(T,B,0)<>0 THEN
  {SPACE} RETURN
JK 2980 IF MAP(T,B,1)=PN+1 THE
  N FQ(0,A)=0:FQ(1,A)=0:
  GOTO3060
XS 2990 J=NX(A):IFJ>31 THEN RETU
  RN
HR 3000 J1=FQ(0,A):IF J1<1 THE
  N 3060
EP 3010 NX(A)=NX(A)+1
XE 3020 MAP(T,B,0)=J:MAP(T,B,1)
  =A+1
BK 3030 ARMY(J,0,A)=J1
QR 3040 FORK=1 TO 3:ARMY(J,K,A)=
  0:NEXTK
PM 3050 ARMY(J,4,A)=T:ARMY(J,5
  ,A)=B
KH 3060 FORK=0 TO 19:FQ(K,A)=FQ(
  K+1,A):NEXTK
PK 3070 FQ(20,A)=0
BA 3080 RETURN
FS 3100 A=0:FORM=1 TO 6:IFRND(1)

```



```

<.5THENA=A+1
AH 3110 NEXTM:B=6-A:RETURN
RD 3200 REM WINDOW
CP 3210 Z=6:PRINT"[7 DOWN]"SPC
(Z)"[7][2 M][2 L][RVS]
[2 K][J][H][G] SCENARI
O [M][N][L][OFF][3 K]
[J][2 G]":GOSUB 3310
XR 3220 PRINTSPC(Z)"[M]
[2 SPACES]1> CAPTURE C
APITAL/FAR[2 SPACES]
[G]"
DB 3230 PRINTSPC(Z)"[M]
[2 SPACES]2> CAPTURE C
APITAL/NEAR [G]"
PF 3240 PRINTSPC(Z)"[M]
[2 SPACES]3> OCCUPY
[3 SPACES]8/12 CITIES
[SPACE][G]"
SF 3250 PRINTSPC(Z)"[M]
[2 SPACES]4> CONTROL
[2 SPACES]6/12 CITIES
[SPACE][G]"
XH 3260 PRINTSPC(Z)"[M]
[2 SPACES]5> OCCUPY
[2 SPACES]40/61 HEXES
[2 SPACES][G]":GOSUB33
10
HX 3270 PRINTSPC(Z)"[2 M][2 L]
[RVS][2 K][J][H][G]
[10 SPACES][M][N][L]
[OFF][3 K][J][2 G]"
MK 3280 POKE198,0:WAIT198,1:GE
TAS:GN=VAL(AS):IFGN<10
RGN>5THEN3280
QK 3290 XC=8:YC=8+GN:GOSUB1:PR
INT"ZZ":FORTD=1TO1000:
NEXT:PRINT"[HOME]"
PX 3300 RETURN
XX 3310 PRINTSPC(Z)"[M]
[26 SPACES][G]":RETURN
GS 3400 A=0:ON GN GOSUB 3430,3
450,3480,3490,3580
JH 3410 IFA=0THENRETURN
QP 3415 QQ=A:EN$=C$:GOSUB600:G
OSUB1710:C$=EN$:A=QQ
MJ 3420 PRINT"[HOME]PLAYER";A;
" WINS":PRINTC$:PRINT
(PRESS ANY KEY)"
SE 3425 POKE198,0:WAIT198,1:RU
N
BB 3430 IF MAP(CIT(2,0),CIT(2,
1),1)=1THENA=2:C$="BLU
E CAPTURED THE CAPITAL
":RETURN
CR 3440 GOTO3460
SG 3450 IF MAP(CIT(3,0),CIT(3,
1),1)=1THENA=2:C$="BLU
E CAPTURED THE CAPITAL
":RETURN
KB 3460 IF MAP(CIT(1,0),CIT(1,
1),1)=2THENA=1:C$="VIO
LET CAPTURED THE CAPIT
AL"
FP 3470 RETURN
BC 3480 L=8:GOTO3500
JB 3490 L=6
FG 3500 C(1)=0:C(2)=0
PK 3510 FORJ=1TO12:T=CIT(J,0):
B=CIT(J,1)
MB 3520 R=MAP(T,B,1):C(R)=C(R)
+1
FB 3530 IF GN=4THEN AN=MAP(T,B
,0)
CX 3535 IFGN<>4 OR R<=0 THEN35
40
HX 3537 IF AN=0 OR ARMY(AN,6,R
-1)>0 THEN C(R)=C(R)-1
KJ 3540 NEXTJ
CB 3550 IF C(1)=> L THEN A=2:C
$="BLUE HAS CAPTURED"+

```

```

STR$(C(1))+ " CITIES":R
ETURN
QF 3560 IF C(2)=> L THEN A=1:C
$="VIOLET HAS CAPTURED
"+STR$(C(2))+ " CITIES"
AB 3570 RETURN
RK 3580 C(1)=0:C(2)=0
PP 3590 FORJ=0TO8:FORK=0TO8
KM 3600 R=MAP(J,K,1):C(R)=C(R)
+1
SR 3610 NEXTK,J
CE 3620 IF C(1)=>40THENA=2:C$=
"BLUE OCCUPIES"+STR$(C
(1))+ " HEXES":RETURN
RK 3630 IF C(2)=>40THENA=1:C$=
"VIOLET OCCUPIES"+STR$
(C(2))+ " HEXES"
DF 3640 RETURN

```

Program 3. Loader for 64 Hex War

```

AG 10 DV=8:Q$=CHR$(34)
DR 20 PRINT"[CLR]POKE 44,64:PO
KE 16384,0:NEW"
XK 30 PRINT"[2 DOWN]LOAD"Q$"HE
X WAR"Q$","DV
QA 40 POKE 198,3:POKE 631,19:P
OKE 632,13:POKE 633,131

```

Program 4. Hex War For Atari 600XL, 800, 800XL, 1200XL and 130XE

Version by Kevin Mykytyn, Editorial Programmer

```

CK 0 PRINT "[CLEAR]":GOTO 9
KP 2 JOY$="QQQQ{D}{B}{C}Q
{F}{H}{G}{Q}{E}{A}{C},":Z=
STICK(0):J=ASC(JOY$(Z,Z
))+128-128*STRIG(0):RET
URN
BM 3 IF QV>255 THEN QV=QV-25
5:GOTO 3
ED 4 H$="0123456789ABCDEF":L
V=INT(QV/16)+1:Z$=H$(LV
,LV):LV=QV-INT(QV/16)*1
6+1:Z$(2,2)=H$(LV,LV):R
ETURN
WM 9 GOSUB 4010:GOSUB 5000:G
OSUB 6000
PC 30 DIM H$(16),Z$(6),JOY$(
16),ARMY(31,14),BTL(64
,8),MAP(9,29),FQ(20,1)
,NX(1),C(2),P$(8),P1$
(8),C$(50),D1$(8)
HB 31 CN=12:DIM CIT(CN,1),D2
$(8),T$(50)
LH 32 POKE 53248,0:HT=0:HB=0
:J=0:K=0:CT=0:CB=0:J1=
0:J2=0:A=0:B=0:C=0
LE 35 FOR A=0 TO 9:FOR B=0 T
O 29:MAP(A,B)=0:NEXT B
:NEXT A:FOR A=0 TO 31:
FOR B=0 TO 14:ARMY(A,B
)=0:NEXT B:NEXT A
LF 37 FOR A=0 TO 8:FOR B=0 T
O 64:BTL(B,A)=0:NEXT B
:NEXT A
BN 60 P0$="#$(DOWN){2 LEFT}
{2 M}":P1$="#{2 N}
(DOWN){2 LEFT}"
BA 70 PN=1:ME=31:SC=704:C4=7
12
AA 80 MM=3:REM MAX MOVES
JI 90 KA=1/48:KB=1/48:KC=1/3
2

```

```

KD 260 RESTORE 270:FOR J=1 T
O CN:FOR K=0 TO 1:REA
D Z:CIT(J,K)=Z:NEXT K
:MAP(CIT(J,0),CIT(J,1
))+10*2=1:NEXT J
AH 270 DATA 8,4,0,4,8,0,0,8,
4,0,4,8
AI 280 DATA 5,5,3,3,6,3,2,5,
5,2,3,6
EJ 300 GOSUB 3200:POSITION 1
5,20:PRINT "SETTING U
P"
NN 340 GOSUB 1500
CN 410 POKE 764,255
LD 420 GOSUB 1900:GOSUB 600:
GOSUB 1710
CC 430 POKE C4,54+100*(1-PN)
:GOSUB 800:POKE 53248
,0
HM 440 POKE C4,104:GOSUB 210
0
FI 450 POKE C4,56:GOSUB 2600
BJ 460 POKE C4,0:GOSUB 2100
CD 470 POKE C4,8:GOSUB 2200
OH 480 GOSUB 2900
CH 490 POKE C4,7:GOSUB 3400
BM 500 PN=1-PN
BE 510 GOTO 410
IK 600 POKE C4,1:POKE SC,15:
GRAPHICS 0:POSITION 0
,2:POKE 82,0:POKE 756
,CHBAS:POKE 559,62:PO
KE 54279,CHBAS
IL 610 SETCOLOR 2,0,12:SETCO
LOR 1,0,4:FOR J=1 TO
5:POKE 85,13-2*J:FOR
K=1 TO J+3:PRINT "-.
";NEXT K:PRINT "-.
"
AL 620 POKE 752,1:POKE 85,12
-2*J:FOR K=1 TO J+4:P
RINT "+, +";NEXT K:P
RINT :NEXT J
DJ 630 FOR J=1 TO 5:POKE 85,
J*2:FOR K=1 TO 10-J:P
RINT ". -";NEXT K:P
RINT
FE 640 POKE 85,J*2+1:FOR K=1
TO 9-J:PRINT "+, ";
:NEXT K:PRINT "+, "
CE 650 NEXT J
PD 660 C$="(Q){E}{DOWN}
{2 LEFT}{Z}{C}":D1$="
%&(DOWN){2 LEFT}":D
2$="";{DOWN}{2 LEFT}<
="
FH 670 FOR J=1 TO 12:GOSUB 7
10:NEXT J
JN 680 J=1:IF BN=1 THEN C$=D
1$:GOSUB 710:J=2:C$=D
2$:GOSUB 710
KO 690 IF BN=2 THEN C$=D1$:G
OSUB 710:J=3:C$=D2$:G
OSUB 710
LC 700 POSITION 0,0:RETURN
OF 710 K=CIT(J,0):L=CIT(J,1)
:X=(K-L)*2+19:Y=(12-(
K+L))*2+3
DH 715 POSITION X,Y:PRINT C$
:RETURN
DJ 800 IF NX(PN)<2 THEN RETU
RN
BH 805 HT=4:HB=4:GOSUB 1000:
POKE SC,101
EH 810 MV=0:CT=0:CB=0:PK=0:K
=0
OD 820 FOR J=1 TO NX(PN)-1:I
F (ARMY(J,PN*7)>0) AN
D (ARMY(J,6+PN*7)<1)
THEN K=1:J=NX(PN)-1
LE 830 NEXT J:IF K=0 THEN RE
TURN
PB 840 IF PEEK(764)=12 THEN
RETURN

```



```

JK 850 GOSUB 2: IF J=0 THEN 8
40
OO 855 IF J>=128 THEN POKE 7
7,0:GOTO 1100
HL 857 IF (J/2)-INT(J/2)=0 T
HEN 840
DP 860 J=(J-1)/2: IF (J/2)-IN
T(J/2) THEN B1=HB+J-2
: T1=HT:GOTO 870
EI 865 T1=HT+1-J: B1=HB
CI 870 IF (T1<0) OR (T1>8) T
HEN 840
AF 880 IF (B1<0) OR (B1>8) T
HEN 840
HF 890 S1=T1+B1: IF (S1<4) OR
(S1>12) THEN 840
EK 895 HB=B1: HT=T1: GOSUB 100
0:GOSUB 940
JB 900 QN=MAP(HT,HB): IF QN=0
THEN 840
NI 905 Q1=MAP(HT,HB+10)-1
MH 910 POSITION 0,0:PRINT "
";QN:PRINT "{5 M}"
MF 920 FOR J=0 TO 3:PRINT "
";ARMY(QN,J+Q1*7):NEX
T J
HB 930 GOTO 840
FI 940 POSITION 0,0:FOR Z=1
TO 6:PRINT "
{5 SPACES}":NEXT Z:RE
TURN
JF 1000 SX=120+8*(HT-HB):SY=
248-16*(HT+HB):POKE
53248,SX:POKE 203,SY
:Z=USR(1536)
HB 1020 IF MAP(HT,HB+20)=1 T
HEN POKE SC,72:RETUR
N
GB 1030 POKE SC,101:RETURN
CC 1100 GOSUB 2: IF J<>0 THEN
1100
EN 1110 IF PK=1 THEN 1200
FH 1120 IF ((MAP(HT,HB+10)<>
PN+1) OR (MAP(HT,HB)
=0)) THEN GOTO 810
PC 1130 AN=MAP(HT,HB): IF ARM
Y(AN,6+PN*7)<>0 THEN
810
FA 1140 PK=1:CT=HT:CB=HB:CS=
ARMY(AN,PN*7)
PA 1150 SOUND 1,100,10,10:FO
R A=1 TO 30:NEXT A:S
OUND 1,0,0,0
JN 1160 GOTO 840
MH 1200 J=((HT=CT) AND (HB=C
B))
JJ 1210 IF J AND (MV=0) THEN
810
NJ 1220 IF J AND (MV>0) THEN
1420
NB 1225 AS=ARMY(MAP(HT,HB),P
N*7)
LN 1230 IF ((AS>ME) AND (CS>
ME)) OR ((MAP(HT,HB+
10)-1=1-PN) AND (AS>
0)) THEN 840
IH 1235 DT=ABS(CT-HT):DB=ABS
(CB-HB)
BE 1240 TL=DB+DT: IF NOT ((T
L=1) OR ((CT+CB=HT+H
B) AND (DT=1))) THEN
840
NH 1250 MG=MAP(HT,HB): IF MG=
0 THEN 1300
DH 1260 FOR J=0 TO 3:ARMY(MG
,J+PN*7)=ARMY(MG,J+P
N*7)+ARMY(AN,J+PN*7)
:ARMY(AN,J+PN*7)=0:N
EXT J
AA 1270 ARMY(MG,6+PN*7)=1:MA
P(CT,CB)=0
JB 1280 CS=ARMY(MG,PN*7):AN=
MG: MV=MM+1
NB 1290 GOTO 1380
NB 1300 NB=MAP(HT,HB+10)-1:M
V=MV+1: IF (NB<>PN) T
HEN MV=MV+1
KJ 1310 MAP(CT,CB)=0
DN 1320 MAP(HT,HB)=AN:MAP(HT
,HB+10)=PN+1:ARMY(AN
,4+PN*7)=HT:ARMY(AN
,5+PN*7)=HB
FB 1322 IF MV=MM THEN ARMY(
AN,6+PN*7)=1
JA 1325 K=0:FOR J=-1 TO 1 ST
EP 2:J1=HT+J:J2=HB+J
:J3=HB-J: IF (J1<0) O
R (J1>8) THEN 1340
KN 1327 IF (MAP(J1,HB)<=0) T
HEN 1340
OE 1330 IF (MAP(J1,HB+10)=2-
PN) THEN K=1:J=1:GOT
O 1360
GM 1340 IF (J2<0) OR (J2>8)
THEN 1350
OK 1342 IF (MAP(HT,J2)>0) TH
EN IF (MAP(HT,J2+10)
=2-PN) THEN K=1:J=1:
GOTO 1360
CH 1350 IF (J3<0) OR (J3>8)
OR (J1<0) OR (J1>8)
THEN 1360
GO 1352 IF (MAP(J1,J3)>0) TH
EN IF (MAP(J1,J3+10)
=2-PN) THEN K=1:J=1
FD 1360 NEXT J
GF 1370 IF K=1 THEN ARMY(AN,
6+PN*7)=1: MV=MM+1
A=PN:J=CT:K=CB:C=0:D
=0:GOSUB 1830
HC 1390 J=HT:K=HB:C=CS:D=ARM
Y(AN,6+PN*7):GOSUB 1
830
LL 1400 CT=HT:CB=HB
JJ 1410 IF MV<MM THEN 840
EF 1420 ARMY(AN,6+PN*7)=1:J=
HT:K=HB:C=CS:D=1:GOS
UB 1830
JK 1430 GOTO 810
HC 1500 RESTORE 1540:FOR J=0
TO 1:NX(J)=5:FOR K=
1 TO 4:READ A,B,C
KB 1510 ARMY(K,J*7)=A:ARMY(K
,4+J*7)=B:ARMY(K,5+J
*7)=C:MAP(B,C)=K:MAP
(B,C+10)=J+1
BF 1520 NEXT K:NEXT J
BL 1540 DATA 64,2,8,64,3,7,6
4,5,6,64,6,6
AG 1550 DATA 64,2,2,64,3,2,6
4,5,1,64,6,0
EI 1600 REM REINFORCE
NL 1610 FOR J=0 TO 1:FOR K=0
TO 20:A=INT(RND(1)*
K*3):A=INT(RND(1)*K*
3):FOR L=1 TO 5:A=A+
INT(RND(1)*21-8)
BL 1620 NEXT L: IF A<16 THEN
A=0:GOTO 1630
IJ 1625 Z=A+K*8:A=INT(Z/2)*2
HI 1630 FQ(K,J)=A:NEXT K:NEX
T J
KL 1640 RETURN
IJ 1700 REM ARMIES>MAP
PG 1710 FOR J=0 TO 8:FOR K=0
TO 8
JN 1720 A=MAP(J,K+10): IF A T
HEN A=A-1:GOSUB 1800
BI 1730 NEXT K:NEXT J
NB 1740 FOR A=0 TO 1:E=13+A*
12:F=A*22:DX=2-4*A:D
=0
CG 1750 FOR J=0 TO 8:C=FQ(J,
A):GOSUB 1840
MF 1760 E=E+DX*2: IF J>3 THEN
F=F+DX:E=E-DX
BC 1770 NEXT J:NEXT A
LA 1780 RETURN
DI 1800 B=MAP(J,K)
OE 1810 C=ARMY(B,A*7)
EH 1820 D=ARMY(B,6+A*7)
NA 1830 E=(J-K+10)*2-1:F=(13
-J-K)*2+1
QA 1840 POSITION E,F: IF A TH
EN PRINT P1*7:GOTO 1
850
DO 1845 PRINT P0*7;
BM 1850 IF C=0 THEN RETURN
MF 1860 POSITION E,F+1-A
BE 1870 QV=C:GOSUB 3:PRINT Z
*7;
LP 1880 IF D THEN POSITION E
,F+A:PRINT CHR$(63-A
);CHR$(63-A)
LC 1890 RETURN
BK 1900 SW=0:E=NX(PN)-1: IF E
<1 THEN RETURN
CE 1910 FOR J=1 TO E-1: IF AR
MY(J,PN*7)>=1 THEN 1
970
KB 1920 T=ARMY(J,4+7*PN):B=A
RMY(J,5+PN*7): IF MAP
(T,B)=J THEN MAP(T,B
)=0
CB 1930 FOR K=J TO E:FOR L=0
TO 6:ARMY(K,L+PN*7)
=ARMY(K+1,L+PN*7):AR
MY(K+1,L+PN*7)=0:NEX
T L
IK 1940 T=ARMY(K,4+PN*7):B=A
RMY(K,5+PN*7):MAP(T,
B)=K
FJ 1950 NEXT K
ON 1960 NX(PN)=NX(PN)-1:J=E:
SW=1
MG 1970 NEXT J: IF SW THEN 19
00
EF 2000 FOR J=1 TO E:ARMY(J,
PN*7)=ARMY(J,PN*7)+A
RMY(J,2+PN*7)
LN 2010 ARMY(J,2+PN*7)=ARMY(
J,3+PN*7):ARMY(J,3+P
N*7)=0
JB 2020 ARMY(J,6+PN*7)=0
ED 2030 NEXT J:K=NX(1-PN):FO
R J=1 TO K:ARMY(J,6+
7*(1-PN))=0:NEXT J
AM 2040 GOSUB 2400
BC 2050 IF BP>0 THEN FOR J=0
TO 1:FOR K=1 TO BP:
A=BTL(K,J):ARMY(A,6+
J*7)=ARMY(A,6+J*7)+1
:NEXT K:NEXT J
KI 2060 RETURN
AJ 2100 GOSUB 2400
KD 2110 A=NX(0): IF NX(1)>A T
HEN A=NX(1)
DA 2120 FOR J=0 TO 1:FOR K=1
TO A:ARMY(K,6+J*7)=
0:NEXT K:NEXT J
AN 2130 GOSUB 2050
KH 2140 RETURN
OO 2200 FOR J=0 TO 1:A=1-J:B
=NX(J)-1
EK 2210 FOR K=1 TO B
KI 2220 IF ARMY(K,J*7)>=1 TH
EN 2280
DD 2230 FQ(2,A)=FQ(2,A)+ARMY
(K,2+J*7)+ARMY(K,3+J
*7): IF FQ(2,A)>255 T
HEN C=FQ(2,A)-255:FQ
(3,A)=FQ(3,A)+C:FQ(2
,A)=255
BK 2240 FQ(6,A)=FQ(6,A)+ARMY
(K,1+J*7): IF FQ(6,A)
>255 THEN C=FQ(6,A)-
255:FQ(7,A)=FQ(7,A)+
C:FQ(6,A)=255
KF 2250 IF (MAP(ARMY(K,4+J*7

```



```

),ARMY(K,5+J*7)=K)
AND (MAP(ARMY(K,4+J*7),
ARMY(K,5+J*7)+10)
)=J+1 THEN 2255
MN 2251 GOTO 2260
KN 2255 MAP(ARMY(K,4+J*7),AR
MY(K,5+J*7))=0
NC 2260 FOR L=0 TO 6:ARMY(K,
L+J*7)=0:NEXT L
AK 2280 IF ARMY(K,6+J*7)>=1
THEN 2320
BA 2290 FQ(4,J)=FQ(4,J)+ARMY
(K,1+J*7):ARMY(K,1+J
*7)=0
HK 2300 IF FQ(4,J)>255 THEN
C=FQ(4,J)-255:FQ(5,J
)=FQ(5,J)+C:FQ(4,J)=
255
CO 2320 NEXT K:NEXT J:RETURN
HF 2400 BP=0
LL 2410 FOR J=0 TO 8:J1=(J-4
)*-(4-J)>0:J2=8+(J>4
)*-(4-J):FOR K=J1 TO
J2
DB 2420 A=MAP(J,K)
NE 2430 R=MAP(J,K+10)
GF 2440 IF A=0 OR R=0 THEN 2
490
BD 2450 IF ARMY(A,(R-1)*7)<1
THEN 2490
II 2460 T=J+1:B=K:GOSUB 2500
BN 2470 B=B-1:GOSUB 2500
JC 2480 T=T-1:GOSUB 2500
DB 2490 NEXT K:NEXT J:RETURN
DI 2500 IF T<0 OR B<0 OR T>8
OR B>8 THEN RETURN
FN 2510 PA=MAP(T,B):IF PA=0
THEN RETURN
HD 2520 IF MAP(T,B+10)=R THE
N RETURN
HP 2530 IF ARMY(PA,(2-R)*7)<
1 THEN RETURN
KF 2540 BP=BP+1:BTL(BP,R-1)=
A:BTL(BP,2-R)=PA:RET
URN
BF 2600 IF BP=0 THEN RETURN
JN 2610 FOR J=1 TO BP
JO 2620 FOR K=0 TO 1:A=1-K
IL 2630 AN=BTL(J,K)
PI 2640 AS=ARMY(AN,K*7):HT=A
RMY(AN,6+K*7):CT=INT
(AS/HT)+1
DB 2650 BTL(J,A+2)=INT(CT*K
A+1)
DK 2660 BTL(J,A+4)=INT(CT*K
B+1)
DO 2670 BTL(J,A+6)=INT(CT*K
C+1)
BN 2680 NEXT K:NEXT J
DD 2700 FOR J=1 TO BP:J0=BTL
(J,0):J1=BTL(J,1)
AD 2710 GOSUB 3100
DL 2720 ARMY(J0,0)=ARMY(J0,0
)-A*BTL(J,2)
EO 2730 ARMY(J1,7)=ARMY(J1,7
)-B*BTL(J,3)
BB 2740 GOSUB 3100
OE 2750 C=A*BTL(J,4):ARMY(J0
,0)=ARMY(J0,0)-C:ARM
Y(J0,1)=ARMY(J0,1)+C
AH 2760 C=B*BTL(J,5):ARMY(J1
,7)=ARMY(J1,7)-C:ARM
Y(J1,8)=ARMY(J1,8)+C
BE 2770 GOSUB 3100
ON 2780 C=A*BTL(J,6):ARMY(J0
,0)=ARMY(J0,0)-C:ARM
Y(J0,3)=ARMY(J0,3)+C
FO 2790 C=B*BTL(J,7):ARMY(J1
,7)=ARMY(J1,7)-C:ARM
Y(J1,10)=ARMY(J1,10)
+C

```

```

FD 2800 NEXT J
KL 2810 RETURN
CO 2900 A=1-PN:B=0
PJ 2910 FOR J=0 TO 8:FOR K=0
TO 8
FL 2920 IF MAP(J,K+10)=PN+1
THEN B=B+1
BL 2930 NEXT K:NEXT J
EA 2950 FQ(1,PN)=FQ(1,PN)+B
BN 2955 IF FQ(1,PN)>255 THEN
B=FQ(1,PN)-255:FQ(2
,PN)=FQ(2,PN)+B:FQ(1
,PN)=255
EP 2960 T=4:B=PN*8
AL 2970 IF MAP(T,B)<>0 THEN
RETURN
KA 2980 IF MAP(T,B+10)=PN+1
THEN FQ(0,A)=0:FQ(1,
A)=0:GOTO 3060
FH 2990 J=NX(A):IF J>31 THEN
RETURN
KJ 3000 J1=FQ(0,A):IF J1<1 T
HEN 3060
MN 3010 NX(A)=NX(A)+1
MO 3020 MAP(T,B)=J:MAP(T,B+1
0)=A+1
CA 3030 ARMY(J,A*7)=J1
MA 3040 FOR K=1 TO 3:ARMY(J,
K+A*7)=0:NEXT K
BF 3050 ARMY(J,4+A*7)=T:ARMY
(J,5+7*A)=B
EM 3060 FOR K=0 TO 19:FQ(K,A
)=FQ(K+1,A):NEXT K
DO 3070 FQ(20,A)=0
KL 3080 RETURN
BP 3100 A=0:FOR M=1 TO 6:IF
RND(1)<0.5 THEN A=A+
1
MI 3110 NEXT M:B=6-A:RETURN
IB 3200 REM WINDOW
DB 3210 GRAPHICS 0:CLOSE #4:
OPEN #4,12,4,"K":Z=
7:POSITION 16,4:PRIN
T "SCENARIO":POKE 75
2,1
FN 3220 POSITION 9,8:PRINT "
1. CAPTURE CAPITAL/F
AR"
NF 3230 POSITION 9,10:PRINT
"2. CAPTURE CAPITAL/
NEAR"
NA 3240 POSITION 9,12:PRINT
"3. OCCUPY 8/12 CITI
ES"
CA 3250 POSITION 9,14:PRINT
"4. CONTROL 6/12 CIT
IES"
NE 3260 POSITION 9,16:PRINT
"5. OCCUPY 40/61 HEX
ES"
BF 3280 GET #4,A:IF CHR$(A)<
"1" OR CHR$(A)>"5" T
HEN 3280
JB 3290 GN=VAL(CHR$(A)):POSI
TION 7,6+GN*2:PRINT
"(2 .)"
KB 3300 RETURN
BB 3400 A=0:ON GN GOSUB 3430
,3450,3480,3490,3580
BE 3410 IF A=0 THEN RETURN
BP 3415 IF A<>0 THEN T=C%:Q
Q=A:GOSUB 600:GOSUB
1710:C%=T%A:A=QQ
BD 3420 GOSUB 940:POSITION 0
,0:PRINT "PLAYER ";A
;" WINS":PRINT C%
MM 3422 PRINT "PRESS ANY KEY
":POKE 764,255
EI 3424 IF PEEK(764)=255 THE
N 3424
NP 3426 POKE C4,15:GOTO 32
KH 3430 IF MAP(CIT(2,0),CIT(
2,1)+10)=1 THEN A=2:

```

```

C%="BLUE CAPTURED TH
E CAPITAL":RETURN
NB 3440 GOTO 3460
KL 3450 IF MAP(CIT(3,0),CIT(
3,1)+10)=1 THEN A=2:
C%="BLUE CAPTURED TH
E CAPITAL":RETURN
EB 3460 IF MAP(CIT(1,0),CIT(
1,1)+10)=2 THEN A=1:
C%="RED CAPTURED THE
CAPITAL"
KO 3470 RETURN
ML 3480 L=8:GOTO 3500
IP 3490 L=6
BH 3500 C(1)=0:C(2)=0
KB 3510 FOR J=1 TO 12:T=CIT(
J,0):B=CIT(J,1)
HF 3520 R=MAP(T,B+10):C(R)=C
(R)+1
NO 3530 IF GN=4 THEN AN=MAP(
T,B):IF R>0 THEN IF
(AN=0) OR (ARMY(AN,6
+(R-1)*7)>0) THEN C(
R)=C(R)-1
FF 3540 NEXT J
EF 3550 IF C(1)>=L THEN A=2:
C%="BLUE HAS CAPTURE
D "
PJ 3560 IF C(2)>=L THEN A=1:
C%="RED HAS CAPTURED
"
JF 3565 IF A THEN Z=LEN(STR$
(C(3-A))):C%(LEN(C%)
+1,LEN(C%)+Z)=STR$(C
(3-A)):C%(LEN(C%)+1,
LEN(C%)+7)=" CITIES"
KP 3570 RETURN
GP 3580 C(1)=0:C(2)=0
PO 3590 FOR J=0 TO 8:FOR K=0
TO 8
HD 3600 R=MAP(J,K+10):C(R)=C
(R)+1
BH 3610 NEXT K:NEXT J
IC 3620 IF C(1)>=40 THEN A=2:
C%="BLUE OCCUPIES "
DB 3630 IF C(2)>=40 THEN A=1:
C%="RED OCCUPIES "
EO 3635 IF A THEN Z=LEN(STR$
(C(3-A))):C%(LEN(C%)
+1,LEN(C%)+Z)=STR$(C
(3-A)):C%(LEN(C%)+1,
LEN(C%)+6)=" HEXES"
KN 3640 RETURN
DD 4010 CHBAS=PEEK(106)-8:PO
KE 106,CHBAS:GRAPHIC
S 0:CHSET=CHBAS*256
NO 4020 POKE 752,1:POSITION
14,10:PRINT "PLEASE
WAIT":FOR A=0 TO 102
3:POKE CHSET+A,PEEK(
57344+A):NEXT A
KH 4070 RESTORE 4080:FOR A=C
HSET+24 TO CHSET+119
:READ B:POKE A,B:NEX
T A
EK 4075 RESTORE 4090:FOR A=C
HSET+208 TO CHSET+23
9:READ B:POKE A,B/2:
NEXT A
LJ 4077 RESTORE 4140:FOR A=C
HSET+240 TO CHSET+24
7:READ B:POKE A+B,B:
POKE A,B*2:NEXT A
PG 4078 FOR A=CHSET+24 TO CH
SET+71:POKE A,PEEK(A
)/2:NEXT A:FOR A=CHS
ET+208 TO CHSET+239:
POKE A,PEEK(A)*2:NEX
T A
LF 4079 FOR A=CHSET+72 TO CH
SET+87:POKE A,PEEK(A
)*2:NEXT A:RETURN
BD 4080 DATA 2,2,10,10,42,42

```



```

,170,170,128,128,160
,160,168,168,170,170
DD 4090 DATA 10,10,10,10,10,
170,170,170,160,160,
160,160,160,170,170,
170
NL 4100 DATA 170,170,170,10,
10,10,10,10,170,170,
170,160,160,160,160,
160
DG 4110 DATA 85,85,21,21,5,5
,1,1,85,85,84,84,80,
80,64,64
KH 4120 DATA 240,240,63,15,3
,3,3,3,15,15,252,240
,192,192,192,192
KI 4130 DATA 3,3,3,3,15,63,2
40,240,192,192,192,1
92,240,252,15,15
EI 4140 DATA 20,20,85,85,85,
20,20,20
CC 5000 POKE 623,1:POKE 5327
7,3:POKE 704,15:POKE
204,CHBAS+4:POKE 53
256,1:RETURN
FO 6000 RESTORE 6000:FOR Z=1
536 TO 1580:READ B:P
OKE Z,B:NEXT Z:RETUR
N
FM 6100 DATA 165,204,133,207
,169,0,133,206,168,1
45,206,136,208,251,1
64,203,162,15,189,29
,6,145,206,200,202,1
6
FL 6110 DATA 247,104,96,255,
129,129,129,129,129,
129,129,129,129,129,
129,129,129,129,255

```

Program 5. Hex War For Apple II

Version by Tim Victor, Editorial Programmer

```

47 10 LOMEM: 24576
B7 20 DIM ARMY(31,6,1),BTL(64,1,
3),MAP(9,9,2),FQ(20,1),NX(
1),C(2)
A2 30 CN = 12: DIM CITY(CN,1)
5D 40 PN = 1:ME = 31:MM = 3:KA =
1 / 48:KB = 1 / 48:KC = 1
/ 32
78 50 FOR J = 1 TO CN: FOR K = 0
TO 1: READ CITY(J,K): NEX
T K:MAP(CITY(J,0),CITY(J,1
),2) = 1: NEXT J
IF 60 GOSUB 1950
D2 70 GOSUB 850
F7 80 GOSUB 2330
F5 90 POKE 6,0: POKE 7,129
F4 100 IF PEEK(190 * 256) = 76
THEN 120
C9 110 POKE 54,0: POKE 55,3: CAL
L 1002: GOTO 130
80 120 PRINT CHR$(4);"PR#A768"
E0 130 A = FRE(0): GOSUB 1110:
GOSUB 220: GOSUB 930
5F 140 GOSUB 340: VTAB 21: HTAB
37: PRINT " ";
DF 150 GOSUB 1250
E3 160 GOSUB 1540
E3 170 GOSUB 1250
F5 180 GOSUB 1290
F1 190 GOSUB 1750
CF 200 GOSUB 2040
58 210 PN = 1 - PN: GOTO 130
23 220 HOME: HGR2: VTAB 3: FOR
J = 1 TO 5: HTAB 13 - 2
* J: FOR K = 1 TO J + 3:

```

```

PRINT "%& ";: NEXT K: PR
INT "%&"
84 230 HTAB 12 - 2 * J: FOR K =
1 TO J + 4: PRINT "%& ";:
NEXT K: PRINT: NEXT J
4E 240 FOR J = 1 TO 5: HTAB 2 *
J: FOR K = 1 TO 10 - J: P
RINT "%& ";: NEXT K: PRI
NT
32 250 HTAB 2 * J + 1: FOR K = 1
TO 9 - J: PRINT "%& ";:
NEXT K: PRINT "%&": NEXT
J
78 260 FOR J = 0 TO 8:H1 = 62 +
14 * ((J - 8) * (J > 4) -
J * (J < 5)):V1 = 27 + 1
6 * J
F5 270 H2 = 265 - H1: HCOLOR= 7:
HPLLOT H1,V1 TO H1,V1 + 9
: HPLLOT H2,V1 TO H2,V1 +
9
6F 280 NEXT J
2E 290 C1$ = "OP":C2$ = "QR": FO
R J = 1 TO 12: GOSUB 330:
NEXT
34 300 C1$ = "ST":C2$ = "UV":J =
1: IF 0N = 1 THEN GOSUB
330:J = 2:C1$ = "WX":C2$
= "YZ": GOSUB 330
28 310 IF 0N = 2 THEN GOSUB 330:
C1$ = "WX":C2$ = "YZ":J =
3: GOSUB 330
17 320 RETURN
99 330 K = CITY(J,0):L = CITY(J,
1):X = (K - L) * 2 + 19:Y
= (12 - (K + L)) * 2 + 4
: VTAB Y: HTAB X: PRINT C
1$: HTAB X: PRINT C2$: RE
TURN
96 340 HT = 4:HB = 4: IF NX(PN)
< 2 THEN RETURN
56 350 MV = 0:CT = 0:CB = 0:PK =
0
2A 355 VTAB 21: HTAB 37: PRINT M
ID$ ("=>?",PN * 2 + 1,2)
;
58 360 K = 0: FOR J = 1 TO NX(PN)
- 1: IF ARMY(J,0,PN) >
0 AND ARMY(J,6,PN) < 1 TH
EN K = 1:J = NX(PN) - 1
C2 370 NEXT J: IF K = 0 THEN RET
URN
C7 380 HC = 4 * (MAP(HT,HB,1) <
2): HCOLOR= 3 + HC: GOSUB
820: GET A$: HCOLOR= HC:
GOSUB 820
97 390 IF A$ = CHR$(13) THEN RE
TURN
19 400 IF A$ = CHR$(3) THEN STO
P
72 410 IF A$ < > " " THEN 710
B9 420 IF PK = 1 THEN 460
9C 430 IF MAP(HT,HB,1) < > PN +
1 OR MAP(HT,HB,0) = 0 THE
N 350
72 440 AN = MAP(HT,HB,0): IF ARM
Y(AN,6,PN) < > 0 THEN 350
23 450 PRINT CHR$(7);:PK = 1:CT
= HT:CB = HB:CS = ARMY(A
N,0,PN): GOTO 380
C4 460 IF (HT < > CT) OR HB < >
CB THEN 490
A6 470 IF MV > 0 THEN 690
A3 480 GOTO 350
D6 490 AS = ARMY(MAP(HT,HB,0),0,
PN): IF AS > ME AND CS >
ME OR MAP(HT,HB,1) = 2 -
PN AND AS > 0 THEN 380
BE 500 DT = ABS(CT - HT):DB = A
BS(CB - HB):TL = DB + DT
: IF TL < > 1 AND (CT + C
B < > HT + HB OR DT < > 1
) THEN 380

```

```

33 510 MG = MAP(HT,HB,0): IF MG
= 0 THEN 560
FA 520 FOR J = 0 TO 3:ARMY(MG,J,
PN) = ARMY(MG,J,PN) + ARM
Y(AN,J,PN):ARMY(AN,J,PN)
= 0: NEXT J
BD 530 ARMY(MG,6,PN) = 1:MAP(CT,
CB,0) = 0
EB 540 CS = ARMY(MG,0,PN):AN = M
G:MV = MM + 1
28 550 GOTO 650
43 560 NB = MAP(HT,HB,1) - 1:MV
= MV + 1: IF NB < > PN TH
EN MV = MV + 1
69 570 MAP(CT,CB,0) = 0:MAP(HT,H
B,0) = AN:MAP(HT,HB,1) =
PN + 1:ARMY(AN,4,PN) = HT
:ARMY(AN,5,PN) = HB: IF M
V > = MM THEN ARMY(AN,6,P
N) = 1
B1 580 K = 0: FOR J = - 1 TO 1 S
TEP 2:J1 = HT + J:J2 = HB
+ J:J3 = HB - J: IF J1 <
0 OR J1 > 8 THEN 600
A3 590 IF MAP(J1,HB,0) > 0 AND M
AP(J1,HB,1) = 2 - PN THEN
K = 1:J = 1: GOTO 640
EE 600 IF J2 < 0 OR J2 > 8 THEN
620
BB 610 IF MAP(HT,J2,0) > 0 AND M
AP(HT,J2,1) = 2 - PN THEN
K = 1:J = 1: GOTO 640
BB 620 IF J3 < 0 OR J3 > 8 OR J1
< 0 OR J1 > 8 THEN 640
2D 630 IF MAP(J1,J3,0) > 0 AND M
AP(J1,J3,1) = 2 - PN THEN
K = 1:J = 1
41 640 NEXT J: IF K = 1 THEN ARM
Y(AN,6,PN) = 1:MV = MM +
1
36 650 A = PN:J = CT:K = CB:C =
0:D = 0: GOSUB 1040
60 660 J = HT:C = HB:C = CS:D =
ARMY(AN,6,PN): GOSUB 1040
CB 670 CT = HT:CB = HB
A6 680 IF MV < MM THEN 380
3E 690 ARMY(AN,6,PN) = 1:J = HT:
K = HB:C = CS:D = 1: GOSU
B 1040
96 700 GOTO 350
C7 710 IF A$ = " " THEN B1 = HB
+ 1:T1 = HT
D3 720 IF A$ = CHR$(21) THEN B1
= HB - 1:T1 = HT
37 730 IF A$ = CHR$(8) THEN T1
= HT - 1:B1 = HB
D1 740 IF A$ = "." THEN T1 = HT
+ 1:B1 = HB
17 750 IF T1 < 0 OR T1 > 8 OR B1
< 0 OR B1 > 8 THEN 380
88 760 S1 = T1 + B1: IF S1 < 4 O
R S1 > 12 THEN 380
84 770 HB = B1:HT = T1:QN = MAP(
HT,HB,0): IF QN = 0 THEN
380
C1 780 Q1 = MAP(HT,HB,1) - 1
D4 790 VTAB 1: HTAB 1:A$ = STR$(
QN): PRINT SPC(4 - LEN
(A$)):A$ = PRINT "-----"
7A 800 FOR J = 0 TO 3:A$ = STR$(
ARMY(QN,J,Q1)): PRINT SP
C(4 - LEN(A$)):A$ = NEXT
J
9C 810 GOTO 380
D2 820 HH = 120 + (HT - HB) * 14
:HV = 219 - (HT + HB) * 1
6
82 830 HPLLOT HH,HV TO HH + 12,HV
- 6 TO HH + 25,HV + 1 TO
HH + 25,HV + 9 TO HH + 1
3,HV + 15 TO HH,HV + 8 TO
HH,HV
28 840 RETURN

```



```

DC 850 FOR J = 0 TO 1: NX(J) = 5:
  FOR K = 1 TO 4: READ A, B, C
8C 860 ARMY(K, 0, J) = A: ARMY(K, 4, J) = B: ARMY(K, 5, J) = C: MAP(B, C, 0) = K: MAP(B, C, 1) = J + 1
36 870 NEXT K, J
45 880 FOR J = 0 TO 1: FOR K = 0 TO 20
55 890 A = INT ( RND (1) * K * 3 ): FOR L = 1 TO 5: A = A + INT ( RND (1) * 21 - 8 ): NEXT L: IF A < 16 THEN A = 0: GOTO 910
F2 900 A = (A + K * 8): A = 2 * INT (A / 2)
8D 910 FQ(K, J) = A: NEXT K, J
1D 920 RETURN
89 930 FOR J = 0 TO 8: FOR K = 0 TO 8
A2 940 A = MAP(J, K, 1): IF (A) THEN A = A - 1: GOSUB 1010
33 950 NEXT K, J
32 960 FOR A = 0 TO 1: E = 13 + A * 12: F = A * 22 + 1: DX = 2 - 4 * A: D = 0
2A 970 FOR J = 0 TO 8: C = FQ(J, A): VTAB F: HTAB E: GOSUB 1050
DC 980 E = E + DX * 2: IF J > 3 THEN F = F + DX: E = E - DX
A8 990 NEXT J, A
D1 1000 RETURN
9D 1010 B = MAP(J, K, 0)
82 1020 C = ARMY(B, 0, A)
D6 1030 D = ARMY(B, 6, A)
3F 1040 GOSUB 1100
A4 1050 PRINT MID$ ("<=> ", A * 2 + 1, 2)
88 1060 HTAB E: PRINT MID$ ("()> ?", A * 2 + 1, 2): IF C = 0 THEN 1090
67 1070 H$ = "": X = C: FOR L = 0 TO 1: T = X: X = INT (X / 16): T = T - 16 * X: H$ = CHR$ (T + 48 + 7 * (T > 9)) + H$: NEXT
25 1080 VTAB F + 1 - A: HTAB E: PRINT H$: IF D THEN VTA B F + A: HTAB E: PRINT MID$ (" - . / ", A * 2 + 1, 2)
83 1090 NORMAL : RETURN
68 1100 E = (J - K + 10) * 2 - 1: F = (13 - J - K) * 2 + 2: HTAB E: VTAB F: RETURN
C8 1110 SW = 0: E = NX(PN) - 1: IF E < 1 THEN RETURN
11 1120 FOR J = 1 TO E - 1: IF ARMY(J, 0, PN) > 0 THEN 1170
D8 1130 T = ARMY(J, 4, PN): B = ARMY(J, 5, PN): IF MAP(T, B, 0) = J THEN MAP(T, B, 0) = 0
1E 1140 FOR K = J TO E: FOR L = 0 TO 6: ARMY(K, L, PN) = ARMY(K + 1, L, PN): ARMY(K + 1, L, PN) = 0: NEXT L
89 1150 T = ARMY(K, 4, PN): B = ARMY(K, 5, PN): MAP(T, B, 0) = K: NEXT K
8A 1160 NX(PN) = NX(PN) - 1: J = E: SW = 1
18 1170 NEXT J: IF SW THEN 1110
66 1180 FOR J = 1 TO E: ARMY(J, 0, PN) = ARMY(J, 0, PN) + ARMY(J, 2, PN)
83 1190 ARMY(J, 2, PN) = ARMY(J, 3, PN): ARMY(J, 3, PN) = 0
69 1200 ARMY(J, 6, PN) = 0
ED 1210 NEXT J: K = NX(1 - PN): F OR J = 1 TO K: ARMY(J, 6, 1 - PN) = 0: NEXT
47 1220 GOSUB 1400
82 1230 IF BP > 0 THEN FOR J = 0 TO 1: FOR K = 1 TO BP: A = BTL(K, J, 0): ARMY(A, 6, J) = ARMY(A, 6, J) + 1: NEXT K, J
E5 1240 RETURN
53 1250 GOSUB 1400
13 1260 A = NX(0): IF (NX(1) > A) THEN A = NX(1)
88 1270 FOR J = 0 TO 1: FOR K = 1 TO A: ARMY(K, 6, J) = 0: NEXT K, J
A8 1280 GOSUB 1230: RETURN
36 1290 FOR J = 0 TO 1: A = 1 - J: B = NX(J) - 1
81 1300 FOR K = 1 TO B
8D 1310 IF ARMY(K, 0, J) > 1 THEN 1360
54 1320 FQ(2, A) = FQ(2, A) + ARMY(K, 2, J) + ARMY(K, 3, J): IF FQ(2, A) > 255 THEN C = FQ(2, A) - 255: FQ(3, A) = FQ(3, A) + C: FQ(2, A) = 255
2E 1330 FQ(6, A) = FQ(6, A) + ARMY(K, 1, J): IF FQ(6, A) > 255 THEN C = FQ(6, A) - 255: FQ(7, A) = FQ(7, A) + C: FQ(6, A) = 255
D2 1340 T = ARMY(K, 4, J): B = ARMY(K, 5, J): IF MAP(T, B, 0) = K AND MAP(T, B, 1) = J + 1 THEN MAP(T, B, 0) = 0
CC 1350 FOR L = 0 TO 6: ARMY(K, L, J) = 0: NEXT L
ED 1360 IF ARMY(K, 6, J) > 1 THEN 1390
88 1370 FQ(4, J) = FQ(4, J) + ARMY(K, 1, J): ARMY(K, 1, J) = 0
44 1380 IF FQ(4, J) > 255 THEN C = FQ(4, J) - 255: FQ(5, J) = FQ(5, J) + C: FQ(4, J) = 255
68 1390 NEXT K, J: RETURN
71 1400 BP = 0
3D 1410 FOR J = 0 TO 8: J1 = (4 - J) * (4 - J > 0): J2 = 8 - (J > 4) * (J - 4)
1B 1420 FOR K = J1 TO J2: A = MAP(J, K, 0): R = MAP(J, K, 1)
8E 1430 IF (A = 0) OR (R = 0) THEN 1480
A8 1440 IF ARMY(A, 0, R - 1) < 1 THEN 1480
D9 1450 T = J + 1: B = K: GOSUB 1490
18 1460 B = B - 1: GOSUB 1490
C1 1470 T = T - 1: GOSUB 1490
66 1480 NEXT K, J: RETURN
31 1490 IF T < 0 OR B < 0 OR T > 8 OR B > 8 THEN RETURN
48 1500 PA = MAP(T, B, 0): IF PA = 0 THEN RETURN
1E 1510 IF MAP(T, B, 1) = R THEN RETURN
DE 1520 IF ARMY(PA, 0, 2 - R) < 1 THEN RETURN
A2 1530 BP = BP + 1: BTL(BP, R - 1, 0) = A: BTL(BP, 2 - R, 0) = PA: RETURN
CC 1540 IF BP = 0 THEN RETURN
33 1550 FOR J = 1 TO BP
6D 1560 FOR K = 0 TO 1: A = 1 - K
21 1570 AN = BTL(J, K, 0)
7D 1580 AS = ARMY(AN, 0, K): HT = ARMY(AN, 6, K): CT = INT (AS / HT) + 1
72 1590 BTL(J, A, 1) = INT (CT * K A + 1)
E8 1600 BTL(J, A, 2) = INT (CT * K B + 1)
75 1610 BTL(J, A, 3) = INT (CT * K C + 1)
86 1620 NEXT K, J
62 1630 FOR J = 1 TO BP: J0 = BTL(J, 0, 0): J1 = BTL(J, 1, 0)
83 1640 GOSUB 1930
1E 1650 ARMY(J0, 0, 0) = ARMY(J0, 0, 0) - A * BTL(J, 0, 1)
6F 1660 ARMY(J1, 0, 1) = ARMY(J1, 0, 1) - B * BTL(J, 1, 1)
8F 1670 GOSUB 1930
A1 1680 C = A * BTL(J, 0, 2): ARMY(J0, 0, 0) = ARMY(J0, 0, 0) - C: ARMY(J0, 1, 0) = ARMY(J0, 1, 0) + C
F2 1690 C = B * BTL(J, 1, 2): ARMY(J1, 0, 1) = ARMY(J1, 0, 1) - C: ARMY(J1, 1, 1) = ARMY(J1, 1, 1) + C
75 1700 GOSUB 1930
A9 1710 C = A * BTL(J, 0, 3): ARMY(J0, 0, 0) = ARMY(J0, 0, 0) - C: ARMY(J0, 3, 0) = ARMY(J0, 3, 0) + C
FA 1720 C = B * BTL(J, 1, 3): ARMY(J1, 0, 1) = ARMY(J1, 0, 1) - C: ARMY(J1, 3, 1) = ARMY(J1, 3, 1) + C
86 1730 NEXT J
EF 1740 RETURN
F5 1750 A = 1 - PN: B = 0
CB 1760 FOR J = 0 TO 8: FOR K = 0 TO 8
AA 1770 IF MAP(J, K, 1) = PN + 1 THEN B = B + 1
28 1780 NEXT K, J
75 1790 FQ(1, PN) = FQ(1, PN) + B: IF FQ(1, PN) > 255 THEN B = FQ(1, PN) - 255: FQ(2, PN) = FQ(2, PN) + B: FQ(1, PN) = 255
84 1800 T = 4: B = PN * 8
27 1810 IF MAP(T, B, 0) < 0 THEN RETURN
8C 1820 IF MAP(T, B, 1) = PN + 1 THEN FQ(0, A) = 0: FQ(1, A) = 0: RETURN
EC 1830 J = NX(A): IF J > 31 THEN RETURN
DE 1840 J1 = FQ(0, A): IF J1 < 1 THEN 1900
6E 1850 NX(A) = NX(A) + 1
8D 1860 MAP(T, B, 0) = J: MAP(T, B, 1) = A + 1
4F 1870 ARMY(J, 0, A) = J1
86 1880 FOR K = 1 TO 3: ARMY(J, K, A) = 0: NEXT K
84 1890 ARMY(J, 4, A) = T: ARMY(J, 5, A) = B
42 1900 FOR K = 0 TO 19: FQ(K, A) = FQ(K + 1, A): NEXT K
EB 1910 FQ(20, A) = 0
EB 1920 RETURN
FC 1930 A = 0: FOR M = 1 TO 6: IF RND (1) < .5 THEN A = A + 1
56 1940 NEXT M: B = 6 - A: RETURN
83 1950 HOME : TEXT : VTAB 6: HTAB 10: PRINT "1> CAPTURE CAPITAL/FAR"
88 1960 HTAB 10: PRINT "2> CAPTURE CAPITAL/NEAR"
73 1970 HTAB 10: PRINT "3> OCCUP Y 8/12 CITIES"
78 1980 HTAB 10: PRINT "4> CONTR OL 6/12 CITIES"
88 1990 HTAB 10: PRINT "5> OCCUP Y 40/61 HEXES"
79 2000 VTAB 12: HTAB 10: PRINT "PRESS KEY TO SELECT GAM E"

```



```

B4 2010 GET A$: IF A$ < "0" OR A
  $ > "5" THEN 2010
B5 2020 GN = VAL (A$): VTAB 5 +
  GN: HTAB 9: PRINT ">>";
B6 2030 VTAB 12: HTAB 4: PRINT "
  PREPARING GAME- ONE MOMEN
  NT PLEASE": RETURN
C7 2040 EA = 0: ON GN GOSUB 2110
  ,2130,2160,2170,2260
B1 2050 IF EA = 0 THEN RETURN
B2 2060 GOSUB 220: GOSUB 930
B3 2070 TEXT : HOME
B5 2080 HTAB 13: PRINT "PLAYER "
  EA" WINS"
A4 2090 HTAB 20 - ( LEN (EN$) -
  .5) / 2: PRINT EN$
B6 2100 HTAB 12: PRINT "(PRESS A
  NY KEY)": CALL 856: GET
  A$: RUN
A6 2110 IF MAP(CITY(2,0),CITY(2,
  1),1) = 1 THEN EA = 2:EN
  $ = "BLUE CAPTURED THE C
  APITAL": RETURN
B8 2120 GOTO 2140
B1 2130 IF MAP(CITY(3,0),CITY(3,
  1),1) = 1 THEN EA = 2:EN
  $ = "BLUE CAPTURED THE C
  APITAL": RETURN
C7 2140 IF MAP(CITY(1,0),CITY(1,
  1),1) = 2 THEN EA = 1:EN
  $ = "VIOLET CAPTURED THE
  CAPITAL"
E8 2150 RETURN
A7 2160 L = 8: GOTO 2180
A8 2170 L = 6
A9 2180 C(1) = 0:C(2) = 0
B1 2190 FOR J = 1 TO 12:T = CITY
  (J,0):B = CITY(J,1)
E9 2200 R = MAP(T,B,1):C(R) = C
  (R) + 1
B2 2210 IF GN = 4 THEN AN = MAP
  (T,B,0): IF R > 0 THEN IF
  AN = 0 OR ARMY(AN,6,R -
  1) > 0 THEN C(R) = C(R)
  - 1
B3 2220 NEXT J
B4 2230 IF C(1) = > L THEN EA =
  2:EN$ = "BLUE HAS CAPTUR
  ED " + STR$ (C(1)) + " C
  ITIES": RETURN
E9 2240 IF C(2) = > L THEN EA =
  1:EN$ = "VIOLET HAS CAPT
  URED " + STR$ (C(2)) + "
  CITIES"
E8 2250 RETURN
B5 2260 C(1) = 0:C(2) = 0
B6 2270 FOR J = 0 TO 8: FOR K =
  0 TO 8
B7 2280 R = MAP(J,K,1):C(R) = C
  (R) + 1
B8 2290 NEXT K,J
B9 2300 IF C(1) = > 40 THEN EA =
  2:EN$ = "BLUE OCCUPIES
  " + STR$ (C(1)) + " HEXE
  S": RETURN
B1 2310 IF C(2) = > 40 THEN EA =
  1:EN$ = "VIOLET OCCUPIE
  S " + STR$ (C(2)) + " HE
  XES"
E8 2320 RETURN
B2 2330 IF PEEK (768) = 216 THEN
  RETURN
B3 2340 READ AD: IF AD = - 1 THE
  N RETURN
B4 2350 READ DT: IF DT = - 1 THE
  N 2340
B5 2360 POKE AD,DT:AD = AD + 1:
  GOTO 2350
B6 2370 DATA 8,4,0,4,8,0,0,8,4,0
  ,4,8
B7 2380 DATA 5,5,3,3,6,3,2,5,5,2
  ,3,6
B8 2390 DATA 64,2,8,64,3,7,64,5,

```

```

6,64,6,6
B9 2400 DATA 64,2,2,64,3,2,64,5,
  1,64,6,0
C1 2410 DATA 33024
B2 2420 DATA 0,0,0,0,0,0,0,0
B3 2430 DATA -1,33048
B4 2440 DATA 131,140,176,192,192
  ,192,192,192
B5 2450 DATA 224,152,134,129,129
  ,129,129,129
B6 2460 DATA 192,192,192,192,192
  ,176,140,131
C2 2470 DATA 129,129,129,129,129
  ,134,152,224
B7 2480 DATA -1,33088
B8 2490 DATA 42,42,0,0,0,0,0,0
B9 2500 DATA 85,85,0,0,0,0,0,0
B1 2510 DATA 0,0,0,0,0,0,170,170
B2 2520 DATA 0,0,0,0,0,0,213,213
B3 2530 DATA 128,128,213,221,255
  ,255,221,213
B4 2540 DATA 128,128,170,174,191
  ,191,174,170
B5 2550 DATA 85,93,127,127,93,85
  ,0,0
B6 2560 DATA 42,46,63,63,46,42,0
  ,0
C3 2570 DATA 128,188,230,246,238
  ,230,188,128
B7 2580 DATA 128,152,156,152,152
  ,152,188,128
B8 2590 DATA 128,188,230,176,140
  ,230,254,128
B9 2600 DATA 128,188,230,176,224
  ,230,188,128
B1 2610 DATA 128,176,184,180,254
  ,176,176,128
B2 2620 DATA 128,254,134,190,224
  ,230,188,128
B3 2630 DATA 128,188,134,190,230
  ,230,188,128
B4 2640 DATA 128,254,224,176,152
  ,140,140,128
B5 2650 DATA 128,188,230,188,230
  ,230,188,128
B6 2660 DATA 128,188,230,230,252
  ,176,152,128
B7 2670 DATA -1,33248
B8 2680 DATA 192,192,208,208,212
  ,212,213,213
B9 2690 DATA 128,128,130,130,138
  ,138,170,170
B1 2700 DATA 85,85,84,84,80,80,6
  4,64
B2 2710 DATA 42,42,10,10,2,2,0,0
B3 2720 DATA -1,33288
B4 2730 DATA 128,252,230,230,254
  ,230,230,128
B5 2740 DATA 128,190,230,230,190
  ,230,254,128
B6 2750 DATA 128,188,230,134,134
  ,230,190,128
B7 2760 DATA 128,190,230,230,230
  ,230,190,128
B8 2770 DATA 128,254,134,134,190
  ,134,254,128
B9 2780 DATA 128,254,134,134,190
  ,134,134,128
B1 2790 DATA -1,33400
B2 2800 DATA 128,128,128,128,224
  ,224,224,252
B3 2810 DATA 128,128,128,128,129
  ,129,129,143
B4 2820 DATA 252,224,224,224,128
  ,128,128,128
B5 2830 DATA 143,129,129,129,128
  ,128,128,128
B6 2840 DATA 128,128,144,144,144
  ,149,149,128
B7 2850 DATA 128,128,130,130,130
  ,170,170,128
B8 2860 DATA 128,149,149,144,144
  ,144,128,128

```

```

B9 2870 DATA 128,170,170,130,130
  ,130,128,128
B1 2880 DATA 0,0,16,16,16,21,21,
  0
B2 2890 DATA 0,0,2,2,2,42,42,0
B3 2900 DATA 0,21,21,16,16,16,0,
  0
B4 2910 DATA 0,42,42,2,2,2,0,0
B5 2920 DATA -1,33528
B6 2930 DATA 0,0,0,0,127,0,0,0
B7 2940 DATA -1,768
B8 2950 DATA 216,120,133,69,134,
  70,132,71
B9 2960 DATA 166,7,10,10,176,4,1
  6,62
B1 2970 DATA 48,4,16,1,232,232,1
  0,134
B2 2980 DATA 27,24,101,6,133,26,
  144,2
B3 2990 DATA 230,27,165,40,133,8
  ,165,41
B4 3000 DATA 41,3,5,230,133,9,16
  2,8
B5 3010 DATA 160,0,177,26,36,50,
  48,2
B6 3020 DATA 73,127,164,36,145,8
  ,230,26
B7 3030 DATA 208,2,230,27,165,9,
  24,105
B8 3040 DATA 4,133,9,202,208,226
  ,165,69
B9 3050 DATA 166,70,164,71,88,76
  ,240,253
B1 3060 DATA 169,120,133,254,169
  ,64,133,255
B2 3070 DATA 169,15,160,7,145,25
  4,136,16
B3 3080 DATA 251,44,85,192,44,80
  ,192,169
B4 3090 DATA 15,205,176,192,208,
  251,44,84
B5 3100 DATA 192,44,81,192,160,2
  51,162,72
B6 3110 DATA 232,208,253,200,208
  ,250,44,0
B7 3120 DATA 192,16,222,96
B8 3130 DATA -1,-1

```

Program 6. Hex War For IBM PC/PCjr

Version by Patrick Parrish,
Programming Supervisor

```

HK 0 KEY OFF:WIDTH 40:DEF SEG=0:
  POKE 1047,PEEK(1047) OR 64:
  SCREEN 1:COLOR 0,1:CLS:LOCA
  TE 12,15,0:PRINT "PLEASE WA
  IT":GOTO 20
A8 2 DEF SEG=0:POKE 1050,PEEK(10
  52)
FP 3 QV=0:A$=RIGHT$(INKEY$,1):IF
  LEN(A$)=0 THEN 3
IP 4 IF A$=CHR$(77) THEN QV=3:GO
  TO 8
NK 5 IF A$=CHR$(75) THEN QV=7:GO
  TO 8
AC 6 IF A$=CHR$(72) THEN QV=1:GO
  TO 8
OF 7 IF A$=CHR$(80) THEN QV=5
HP 8 J=QV-128*(A$=" "):RETURN
CF 20 GOSUB 3710
CI 40 DIM ARMY(31,6,1),BTL(64,1,
  3),MAP(9,9,2),FQ(20,1),NX(1,
  1),C(2)
EB 50 CN=12:DIM CIT(CN,1)
HP 60 RANDOMIZE (TIMER)
EJ 70 PN=1:ME=31
JC 80 MM=3:REM MAX MOVES
FF 90 KA=1/48:KB=1/48:KC=1/32
KB 260 RESTORE 270:FOR J=1 TO CN
  :FOR K=0 TO 1:READ CIT(J,

```



```

K):NEXT K:MAP(CIT(J,0),CI
T(J,1),2)=1:NEXT J
DC 270 DATA 8,4,0,4,8,0,0,8,4,0,
4,8
DM 280 DATA 5,5,3,3,6,3,2,5,5,2,
3,6
LK 300 GOSUB 600:GOSUB 3200:GOSU
B 600
JH 310 LOCATE 24,17:PRINT "HEX W
AR";
QI 330 LOCATE 1,11:PRINT "PLEASE
WAIT A MOMENT"
BO 340 GOSUB 1500
IH 410 REM CLEAR KEYBD
CH 420 GOSUB 1900:GOSUB 600:GOSU
B 1710:REM FIND BATTLES
DO 430 IF PN=1 THEN PUT (280,160
),S5,PSET ELSE PUT (280,1
60),S6,PSET
FO 431 GOSUB 800:LOCATE 21,36:PR
INT " ":REM KEYBD
CL 440 COLOR 14:GOSUB 2100:REM B
ATTLES AGAIN
NL 450 COLOR 1:GOSUB 2600:REM RE
SOLVE
HI 460 COLOR 1:GOSUB 2100:REM PO
ST-BATTLE
GB 470 COLOR 2:GOSUB 2200:REM SP
LIT PRISONERS
EM 480 GOSUB 2900:REM REINFORCEM
ENTS
EM 490 COLOR 6:GOSUB 3400
OK 500 PN=1:PN:FT=0:PP=0
DB 510 GOTO 420
PF 600 CLS:COLOR 0
KJ 610 FOR R=11 TO 1 STEP -2:FOR
C=12-R TO R+26 STEP 4:PU
T (C*8,R*8),S10:NEXT C,R
FM 620 FOR R=13 TO 21 STEP 2:FOR
C=R-10 TO 49-R STEP 4:PU
T (C*8,R*8),S10:NEXT C,R
GN 630 FOR R=12 TO 1 STEP -2:FOR
C=14-R TO R+28 STEP R+28
-(14-R)-1:LOCATE R,C:PRIN
T " ":NEXT C,R
QI 640 FOR R=13 TO 21 STEP 2:FOR
C=R-11 TO 53-R STEP 53-R
-(R-11)-1:LOCATE R,C:PRIN
T " ":NEXT C,R
EH 650 FOR I=2 TO 23 STEP 21:LOC
ATE I,12,0:PRINT "
":NEXT
QO 670 FOR J=1 TO 12:GOSUB 710:N
EXT
BP 680 J=1:IF GN=1 THEN GOSUB 71
8:J=2:GOSUB 715
AH 690 IF GN=2 THEN GOSUB 718:J=
3:GOSUB 715
DB 700 LOCATE 1,1:RETURN
NG 710 K=CIT(J,0):L=CIT(J,1):X=(
K-L)*2+19:Y=(12-(K+L))*2+
3:PUT (X*8+3,Y*8+3),S2,PS
ET:RETURN
KJ 715 K=CIT(J,0):L=CIT(J,1):X=(
K-L)*2+19:Y=(12-(K+L))*2+
3:PUT (X*8,Y*8),S3,PSET:R
ETURN
PK 718 K=CIT(J,0):L=CIT(J,1):X=(
K-L)*2+19:Y=(12-(K+L))*2+
3:PUT (X*8,Y*8),S4,PSET:R
ETURN
JJ 800 IF NX(PN)<2 THEN RETURN
LP 805 HT=4:HB=4:GOSUB 1000
OI 810 MV=0:CT=0:CB=0:PK=0:K=0
IB 820 FOR J=1 TO NX(PN)-1:IF AR
MY(J,0,PN)>0 AND ARMY(J,6
,PN)<1 THEN K=1:J=NX(PN)-
1
JO 830 NEXT J:IF K=0 THEN RETURN
CD 840 IF A$=CHR$(27) THEN A$=""
:RETURN
AH 850 GOSUB 2:IF J=0 THEN 840 E
LSE IF (J AND 128) THEN 1
100 ELSE IF (J AND 1)=0 T
HEN 840
GL 860 J=(J-1)/2:IF J AND 1 THEN
B1=HB+J-2:T1=HT ELSE T1=
HT+1-J:B1=HB
PP 870 IF T1<0 OR T1>8 THEN 840
EB 880 IF B1<0 OR B1>8 THEN 840
FE 890 S1=T1+B1:IF S1<4 OR S1>12
THEN 840
GA 895 HB=B1:HT=T1:GOSUB 1000
DB 896 LOCATE 1,1:FOR Z=1 TO 6:P
RINT " ":NEXT
CJ 900 QN=MAP(HT,HB,0):IF QN=0 T
HEN LOCATE 1,1:GOTO 840 E
LSE Q1=MAP(HT,HB,1)-1
GJ 910 LOCATE 1,1:PRINT USING "#
###";QN:PRINT"-----"
CK 920 FOR J=0 TO 3:PRINT USING
"####";ARMY(QN,J,Q1):NEXT
DI 930 LOCATE 1,1:GOTO 840
NL 1000 SX=150+16*(HT-HB):SY=214
-16*(HT+HB)
CC 1005 IF MAP(HT,HB,2)=1 THEN P
UT (OX,OY),S8:PUT (SX,SY
),S7:OX=SX:OY=SY:PP=1:RE
TURN
AF 1007 IF PP THEN PUT (OX,OY),S
7:PUT (SX,SY),S8:OX=SX:O
Y=SY:PP=0:RETURN
OJ 1010 IF FT THEN PUT (OX,OY),S
8:PUT (SX,SY),S8:OX=SX:O
Y=SY:RETURN
NJ 1015 PUT (SX,SY),S8:OX=SX:OY=
SY:FT=1
IE 1030 RETURN
JF 1100 IF PK=1 THEN 1200
BA 1120 IF MAP(HT,HB,1)<>PN+1 OR
MAP(HT,HB,0)=0 THEN 810
KB 1130 AN=MAP(HT,HB,0):IF ARMY(
AN,6,PN)<>0 THEN 810
PJ 1140 PK=1:CT=HT:CB=HB:CS=ARMY
(AN,0,PN)
NO 1150 SOUND 2200,10
FH 1160 GOTO 840
KK 1200 J=(HT=CT) AND (HB=CB)
ED 1210 IF J AND MV=0 THEN 810
HM 1220 IF J AND MV>0 THEN 1420
NH 1230 AS=ARMY(MAP(HT,HB,0),0,P
N):IF (AS>ME AND CS>ME)
OR (MAP(HT,HB,1)-1=1-PN
AND AS>0) THEN 840
OE 1240 DT=ABS(CT-HT):DB=ABS(CB-
HB):TL=DB+DT:IF NOT (TL=
1 OR (CT+CB=HT+HB AND DT
=1)) THEN 840
CF 1250 MG=MAP(HT,HB,0):IF MG=0
THEN 1300
CL 1260 FOR J=0 TO 3:ARMY(MG,J,P
N)=ARMY(MG,J,PN)+ARMY(AN
,J,PN):ARMY(AN,J,PN)=0:N
EXT J
QM 1270 ARMY(MG,6,PN)=1:MAP(CT,C
B,0)=0
BN 1280 CS=ARMY(MG,0,PN):AN=MG:M
V=MM+1
BB 1290 GOTO 1380
JF 1300 NB=MAP(HT,HB,1)-1:MV=MV+
1:IF NB<>PN THEN MV=MV+1
IL 1310 MAP(CT,CB,0)=0
IN 1320 MAP(HT,HB,0)=AN:MAP(HT,H
B,1)=PN+1:ARMY(AN,4,PN)=
HT:ARMY(AN,5,PN)=HB:IF M
V>MM THEN ARMY(AN,6,PN)
=1
CJ 1330 K=0:FOR J=-1 TO 1 STEP 2
:J1=HT+J:J2=HB+J:J3=HB-J
:IF J1<0 OR J1>8 THEN 13
40 ELSE IF MAP(J1,HB,0)>
0 THEN IF MAP(J1,HB,1)=2
-PN THEN K=1:J=1:GOTO 13
60
QO 1340 IF J2<0 OR J2>8 THEN 135
0 ELSE IF MAP(HT,J2,0)>0
THEN IF MAP(HT,J2,1)=2-
PN THEN K=1:J=1:GOTO 136
0
FB 1350 IF J3<0 OR J3>8 OR J1<0
OR J1>8 THEN 1360 ELSE I
F MAP(J1,J3,0)>0 THEN IF
MAP(J1,J3,1)=2-PN THEN
K=1:J=1
HC 1360 NEXT J
NM 1370 IF K=1 THEN ARMY(AN,6,PN
)=1:MV=MM+1
DL 1380 A=PN:J=CT:K=CB:C=0:D=0:G
OSUB 1830
NF 1390 J=HT:K=HB:C=CS:D=ARMY(AN
,6,PN):GOSUB 1830
NJ 1400 CT=HT:CB=HB
LH 1410 IF MV<MM THEN 840
PJ 1420 ARMY(AN,6,PN)=1:J=HT:K=H
B:C=CS:D=1:GOSUB 1830
DA 1430 GOTO 810
BI 1500 RESTORE 1540:FOR J=0 TO
1:NX(J)=5:FOR K=1 TO 4:R
EAD A,B,C
DA 1510 ARMY(K,0,J)=A:ARMY(K,4,J
)=B:ARMY(K,5,J)=C:MAP(B,
C,0)=K:MAP(B,C,1)=J+1
EH 1520 NEXT K,J
OK 1530 REM STRENGTH, T-POS, B-P
OS
EL 1540 DATA 64,2,8,64,3,7,64,5,
6,64,6,6:REM BLUE
HO 1550 DATA 64,2,2,64,3,2,64,5,
1,64,6,0:REM VIOLET
EG 1600 REM SET RANDOM REINFORCE
MENTS
AH 1610 FOR J=0 TO 1:FOR K=0 TO
20
PH 1620 A=INT(RND(1)*K*3):FOR L=
1 TO 5:A=A+INT(RND(1)*21
-8):NEXT L:IF A<16 THEN
A=0 ELSE A=(A+K*8) AND 2
54
DJ 1630 FQ(K,J)=A:NEXT K,J
JD 1640 RETURN
FF 1700 REM ARMIES->MAP
BE 1710 FOR J=0 TO 8:FOR K=0 TO
8
AH 1720 A=MAP(J,K,1):IF A THEN A
=A-1:GOSUB 1800
FO 1730 NEXT K,J
LI 1740 FOR A=0 TO 1:E=13+A*12:F
=A*22:DX=2-4*A:D=0
AA 1750 FOR J=0 TO 8:C=FQ(J,A):G
OSUB 1840
OM 1760 E=E+DX*2:IF J>3 THEN F=F
+DX:E=E-DX
OK 1770 NEXT J,A
KB 1780 RETURN
HM 1800 B=MAP(J,K,0)
PF 1810 C=ARMY(B,0,A)
BI 1820 D=ARMY(B,6,A)
OF 1830 E=(J-K+10)*2-1:F=(13-J-K
)*2+1:REM T&B TO X/Y
HM 1840 IF A THEN PUT (E*8,F*8),
S5,PSET:LOCATE F+A+1,E+1
:PRINT " "; ELSE PUT (E
*8,(F+1)*8),S6,PSET:LOCA
TE F+A+1,E+1:PRINT " ";
EG 1850 IF C=0 THEN RETURN
LD 1870 LOCATE F+A+1,E+1:PRINT R
IGHT$("0"+HEX$(C),2);
BP 1880 IF D AND A=0 THEN LOCATE
F+2,E+1:PRINT " ";:PUT
(E*8+1,(F+1)*8+1),S11 E
LSE IF D AND A=1 THEN LO
CATE F+1,E+1:PRINT " ";
:PUT (E*8+1,F*8+1),S9
KB 1890 RETURN
HJ 1900 SW=0:E=NX(PN)-1:IF E<1 T
HEN RETURN
JN 1910 FOR J=1 TO E-1:IF ARMY(J
,0,PN)>=1 THEN 1970
HC 1930 FOR K=J TO E:FOR L=0 TO

```



```

6: ARMY(K,L,PN)=ARMY(K+1,
L,PN): ARMY(K+1,L,PN)=0: N
EXT L
CB 1940 T=ARMY(K,4,PN): B=ARMY(K,
5,PN): MAP(T,B,0)=K
IB 1950 NEXT K
IE 1960 NX(PN)=NX(PN)-1: J=E: SW=1
BL 1970 NEXT J: IF SW THEN 1900
GP 2000 FOR J=1 TO E: ARMY(J,0,PN)
=ARMY(J,0,PN)+ARMY(J,2,
PN)
HI 2010 ARMY(J,2,PN)=ARMY(J,3,PN)
: ARMY(J,3,PN)=0
PM 2020 ARMY(J,6,PN)=0
GF 2030 NEXT J: K=NX(1-PN): FOR J=
1 TO K: ARMY(J,6,1-PN)=0:
NEXT
FG 2040 GOSUB 2400
IL 2050 IF BP>0 THEN FOR J=0 TO
1: FOR K=1 TO BP: A=BTL(K,
J,0): ARMY(A,6,J)=ARMY(A,
6,J)+1: NEXT K, J
JO 2060 RETURN
FM 2100 GOSUB 2400
GL 2110 A=NX(0): IF NX(1)>A THEN
A=NX(1)
EN 2120 FOR J=0 TO 1: FOR K=1 TO
A: ARMY(K,6,J)=0: NEXT K, J
GH 2130 GOSUB 2050
JK 2140 RETURN
CO 2200 FOR J=0 TO 1: A=1-J: B=NX(
J)-1
CK 2210 FOR K=1 TO B
OG 2220 IF ARMY(K,0,J)>=1 THEN 2
280
PH 2230 FQ(2,A)=FQ(2,A)+ARMY(K,2,
J)+ARMY(K,3,J): IF FQ(2,
A)>255 THEN C=FQ(2,A)-25
5: FQ(3,A)=FQ(3,A)+C: FQ(2,
A)=255
EP 2240 FQ(6,A)=FQ(6,A)+ARMY(K,1,
J): IF FQ(6,A)>255 THEN
C=FQ(6,A)-255: FQ(7,A)=FQ
(7,A)+C: FQ(6,A)=255
BB 2250 IF MAP(ARMY(K,4,J), ARMY(
K,5,J),0)=K AND MAP(ARMY
(K,4,J), ARMY(K,5,J),1)=J
+1 THEN MAP(ARMY(K,4,J),
ARMY(K,5,J),0)=0
NE 2260 FOR L=0 TO 6: ARMY(K,L,J)
=0: NEXT L
KP 2280 IF ARMY(K,6,J)>=1 THEN 2
320: REM EVACUATE INJURED
PI 2290 FQ(4,J)=FQ(4,J)+ARMY(K,1,
J): ARMY(K,1,J)=0
MO 2300 IF FQ(4,J)>255 THEN C=FQ
(4,J)-255: FQ(5,J)=FQ(5,J)
+C: FQ(4,J)=255
FK 2320 NEXT K, J: RETURN
OO 2400 BP=0
IO 2410 FOR J=0 TO 8: J1=(J-4)*(4
-J)>0: J2=8-(J-4)*(4-J): F
OR K=J1 TO J2
BF 2420 A=MAP(J,K,0)
OO 2430 R=MAP(J,K,1)
JC 2440 IF A=0 OR R=0 THEN 2490
JN 2450 IF ARMY(A,0,R-1)<1 THEN
2490
AG 2460 T=J+1: B=K: GOSUB 2500
CC 2470 B=B-1: GOSUB 2500
CB 2480 T=T-1: GOSUB 2500
GB 2490 NEXT K, J: RETURN
OM 2500 IF T<0 OR B<0 OR T>8 OR
B>8 THEN RETURN
IA 2510 PA=MAP(T,B,0): IF PA=0 TH
EN RETURN
CN 2520 IF MAP(T,B,1)=R THEN RET
URN
CO 2530 IF ARMY(PA,0,2-R)<1 THEN
RETURN
KF 2540 BP=BP+1: BTL(BP,R-1,0)=A:
BTL(BP,2-R,0)=PA: RETURN
AH 2600 IF BP=0 THEN RETURN
NI 2610 FOR J=1 TO BP
ED 2620 FOR K=0 TO 1: A=1-K
EG 2630 AN=BTL(J,K,0)
BH 2640 AS=ARMY(AN,0,K): HT=ARMY(
AN,6,K): CT=INT(AS/HT)+1
MG 2650 BTL(J,A,1)=INT(CT*K+1)
PA 2660 BTL(J,A,2)=INT(CT*K+1)
CK 2670 BTL(J,A,3)=INT(CT*K+1)
BH 2680 NEXT K, J
IB 2700 FOR J=1 TO BP: J0=BTL(J,0,
0): J1=BTL(J,1,0)
EA 2710 GOSUB 3100
DC 2720 ARMY(J0,0,0)=ARMY(J0,0,0)
-A*BTL(J,0,1)
MP 2730 ARMY(J1,0,1)=ARMY(J1,0,1)
-B*BTL(J,1,1)
FJ 2740 GOSUB 3100
CI 2750 C=A*BTL(J,0,2): ARMY(J0,0,
0)=ARMY(J0,0,0)-C: ARMY(
J0,1,0)=ARMY(J0,1,0)+C
MP 2760 C=B*BTL(J,1,2): ARMY(J1,0,
1)=ARMY(J1,0,1)-C: ARMY(
J1,1,1)=ARMY(J1,1,1)+C
FC 2770 GOSUB 3100
DB 2780 C=A*BTL(J,0,3): ARMY(J0,0,
0)=ARMY(J0,0,0)-C: ARMY(
J0,3,0)=ARMY(J0,3,0)+C
OI 2790 C=B*BTL(J,1,3): ARMY(J1,0,
1)=ARMY(J1,0,1)-C: ARMY(
J1,3,1)=ARMY(J1,3,1)+C
HL 2800 NEXT J
JP 2810 RETURN
JE 2900 A=1-PN: B=0
HJ 2910 FOR J=0 TO 8: FOR K=0 TO
8
OA 2920 IF MAP(J,K,1)=PN+1 THEN
B=B+1
FD 2930 NEXT K, J
HD 2950 FQ(1,PN)=FQ(1,PN)+B
CN 2955 IF FQ(1,PN)>255 THEN B=F
Q(1,PN)-255: FQ(2,PN)=FQ(
2,PN)+B: FQ(1,PN)=255
HA 2960 T=4: B=PN*8
MP 2970 IF MAP(T,B,0)<>0 THEN RE
TURN
HH 2980 IF MAP(T,B,1)=PN+1 THEN
FQ(0,A)=0: FQ(1,A)=0: GOTO
3060
NC 2990 J=NX(A): IF J>31 THEN RET
URN
BF 3000 J1=FQ(0,A): IF J1<1 THEN
3060
IA 3010 NX(A)=NX(A)+1
EP 3020 MAP(T,B,0)=J: MAP(T,B,1)=
A+1
HC 3030 ARMY(J,0,A)=J1
EG 3040 FOR K=1 TO 3: ARMY(J,K,A)
=0: NEXT K
HN 3050 ARMY(J,4,A)=T: ARMY(J,5,A)
=B
PF 3060 FOR K=0 TO 19: FQ(K,A)=FQ
(K+1,A): NEXT K
IA 3070 FQ(20,A)=0
JF 3080 RETURN
IC 3100 A=0: FOR M=1 TO 6: IF RND(
1)<.5 THEN A=A+1
KD 3110 NEXT M: B=6-A: RETURN
PO 3200 REM WINDOW
FH 3210 LOCATE 8,7: PRINT STRING$
(9,19): SCENARIO "STRING
$(9,19): LOCATE 9,7: GOSUB
3310
PB 3220 LOCATE 10,7: PRINT CHR$(1
9)" 1> CAPTURE CAPITAL/F
AR "CHR$(19)
PN 3230 LOCATE 11,7: PRINT CHR$(1
9)" 2> CAPTURE CAPITAL/N
EAR "CHR$(19)
NG 3240 LOCATE 12,7: PRINT CHR$(1
9)" 3> OCCUPY 8/12 CIT
IES "CHR$(19)
FA 3250 LOCATE 13,7: PRINT CHR$(1
9)" 4> CONTROL 6/12 CIT
IES "CHR$(19)
LF 3260 LOCATE 14,7: PRINT CHR$(1
9)" 5> OCCUPY 40/61 HEX
ES "CHR$(19): LOCATE 15
,7: GOSUB 3310
IO 3270 LOCATE 16,7: PRINT STRING
$(28,19)
DI 3280 A$=INKEY$: IF A$="" THEN
3280 ELSE GN=VAL(A$): IF
GN<1 OR GN>5 THEN 3280
BH 3290 LOCATE 9+GN,9: PRINT CHR$
(16): FOR TD=1 TO 1000: NE
XT
ID 3300 RETURN
HO 3310 PRINT CHR$(19) SPC(26) CHR
$(19): RETURN
IA 3400 A=0: ON GN GOSUB 3430,345
0,3480,3490,3580
EK 3410 IF A=0 THEN RETURN ELSE
EN$=C$: EA=A: GOSUB 600: GO
SUB 1710: A=EA
HL 3420 LOCATE 1,1: PRINT "PLAYER
"A" WINS
": PRINT EN$: PRINT "(PRE
SS ANY KEY)"
CL 3425 A$=INKEY$: IF A$="" THEN
3425 ELSE RUN
EH 3430 IF MAP(CIT(2,0), CIT(2,1)
,1)=1 THEN A=2: C$="BLUE
CAPTURED THE CAPITAL ": R
ETURN
BA 3440 GOTO 3460
II 3450 IF MAP(CIT(3,0), CIT(3,1)
,1)=1 THEN A=2: C$="BLUE
CAPTURED THE CAPITAL ": R
ETURN
PF 3460 IF MAP(CIT(1,0), CIT(1,1)
,1)=2 THEN A=1: C$="VIOLE
T CAPTURED THE CAPITAL "
KK 3470 RETURN
KL 3480 L=8: GOTO 3500
NJ 3490 L=6
OH 3500 C(1)=0: C(2)=0
HD 3510 FOR J=1 TO 12: T=CIT(J,0)
: B=CIT(J,1)
OI 3520 R=MAP(T,B,1): C(R)=C(R)+1
KB 3530 IF GN=4 THEN AN=MAP(T,B,
0): IF R>0 THEN IF AN=0 O
R ARMY(AN,6,R-1)>0 THEN
C(R)=C(R)-1
HC 3540 NEXT J
NE 3550 IF C(1)>L THEN A=2: C$="
BLUE HAS CAPTURED"+STR$(
C(1))+ " CITIES ": RETURN
NO 3560 IF C(2)>L THEN A=1: C$="
VIOLET HAS CAPTURED"+STR
$(C(2))+ " CITIES "
KH 3570 RETURN
PE 3580 C(1)=0: C(2)=0
IK 3590 FOR J=0 TO 8: FOR K=0 TO
8
OK 3600 R=MAP(J,K,1): C(R)=C(R)+1
FI 3610 NEXT K, J
KH 3620 IF C(1)>40 THEN A=2: C$="
BLUE OCCUPIES"+STR$(C(1)
)+ " HEXES ": RETURN
CA 3630 IF C(2)>40 THEN A=1: C$="
VIOLET OCCUPIES"+STR$(C
(2))+ " HEXES "
JF 3640 RETURN
EE 3700 REM DEFINE SHAPES
DP 3710 DEFINT S
IB 3720 RESTORE 3840: READ X,Y,E=
(4+INT((X+7)/8)*Y)/2: DIM
S10(E): S10(0)=X: S10(1)=
Y: FOR I=2 TO E: READ S10(
I): NEXT
PJ 3730 READ X,Y,E=(4+INT((X+7)/
8)*Y)/2: DIM S2(E): S2(0)=
X: S2(1)=Y: FOR I=2 TO E: R
EAD S2(I): NEXT
NE 3740 READ X,Y,E=(4+INT((X+7)/
8)*Y)/2: DIM S3(E): S3(0)=

```



```

X:S3(1)=Y:FOR I=2 TO E:R
EAD S3(I):NEXT
MP 3750 READ X,Y:E=(4+INT((X+7)/
B)*Y)/2:DIM S4(E):S4(0)=
X:S4(1)=Y:FOR I=2 TO E:R
EAD S4(I):NEXT
LK 3760 READ X,Y:E=(4+INT((X+7)/
B)*Y)/2:DIM S5(E):S5(0)=
X:S5(1)=Y:FOR I=2 TO E:R
EAD S5(I):NEXT
JF 3770 READ X,Y:E=(4+INT((X+7)/
B)*Y)/2:DIM S6(E):S6(0)=
X:S6(1)=Y:FOR I=2 TO E:R
EAD S6(I):NEXT
IA 3780 READ X,Y:E=(4+INT((X+7)/
B)*Y)/2:DIM S7(E):S7(0)=
X:S7(1)=Y:FOR I=2 TO E:R
EAD S7(I):NEXT
HL 3790 READ X,Y:E=(4+INT((X+7)/
B)*Y)/2:DIM S8(E):S8(0)=
X:S8(1)=Y:FOR I=2 TO E:R
EAD S8(I):NEXT
EK 3800 READ X,Y:E=(4+INT((X+7)/
B)*Y)/2:DIM S9(E):S9(0)=
X:S9(1)=Y:FOR I=2 TO E:R
EAD S9(I):NEXT
DA 3810 READ X,Y:E=(4+INT((X+7)/
B)*Y)/2:DIM S11(E):S11(0)=
X:S11(1)=Y:FOR I=2 TO
E:READ S11(I):NEXT
JD 3820 RETURN
HH 3830 REM HEX SHAPE S10
CH 3840 DATA &H20,&H10,&HFC,&H3F
00,&HFC,&H3F00,&HFF03,&H
C0FF
BE 3850 DATA &HFF00,&HFF,&HFF00,&
HF0,&HFF00,&HFF,&HFF00,&H
F0
BF 3860 DATA &HFF00,&HFF0,&HFF00,&H
F0,&HFF00,&HFF0,&HFF00,&H
F0
FB 3870 DATA &HFF00,&HFF0,&HFF00,&
HFF,&HFF03,&HFC0FF,&HFC,&
H3F00
ED 3880 DATA &HFC,&H3F00,&H0
AD 3890 REM CITY SHAPE S2
PL 3900 DATA &H14,&HA,&HAA0A,&H2
A00,&HB0AA,&HA0,&HAA00,&
HA000
OB 3910 DATA &HA0,&HAA0A,&HAA000,&
HA0,&HAA0A,&HAA000,&HAA2
A,&HAB0
CM 3920 DATA &HAA,&H0
HC 3930 REM CAP.PUR SHAPE S3
IN 3940 DATA &H20,&H10,&HAA00,&H
AA,&HAA00,&HAA,&HAA00,&H
AA
DI 3950 DATA &HAA00,&HAA,&HAAAA,&
HAAAA,&HAAAA,&HAAAA,&HAA
AAA,&HAAAA
BP 3960 DATA &HAAAA,&HAAAA,&HAAA
A,&HAAAA,&HAAAA,&HAAAA,&
HAAAA,&HAAAA
EO 3970 DATA &HAAAA,&HAAAA,&HAA0
0,&HAA,&HAA00,&HAA,&HAA0
0,&HAA
ON 3980 DATA &HAA00,&HAA,&H0
IC 3990 REM CAP.BLU SHAPE S4
HA 4000 DATA &H20,&H10,&H5500,&H
55,&H5500,&H55,&H5500,&H
55
GL 4010 DATA &H5500,&H55,&H5555,&
H5555,&H5555,&H5555,&H5
5555
HC 4020 DATA &H5555,&H5555,&H5555
5,&H5555,&H5555,&H5555,&
H5555
DB 4030 DATA &H5555,&H5555,&H550
0,&H55,&H5500,&H55,&H550
0,&H55
JA 4040 DATA &H5500,&H55,&H0
ON 4050 REM TOP.PUR SHAPE S5
CM 4060 DATA &H20,&H0,&H200,&H00
,&HA00,&HA0,&H2A00,&HAB

```

```

CH 4070 DATA &HAA00,&HAA,&HAA02,&
H80AA,&HAA0A,&HAA0A,&HA
A2A,&HABAA
GC 4080 DATA &H0,&H0,&H0
OH 4090 REM TOP.BLU SHAPE S6
FF 4100 DATA &H20,&H0,&H5515,&H5
455,&H5505,&H55055,&H5501
,&H4055
DE 4110 DATA &H5500,&H55,&H1500,&
H54,&H500,&H50,&H100,&H
40
FC 4120 DATA &H0,&H0,&H0
DC 4130 REM CURSOR.PUR SHAPE S7
EK 4140 DATA &H20,&H14,&H2A00,&H
ABAA,&H0,&HAA2A,&HAB,&H0
CN 4150 DATA &H0,&H0,&H0,&H0,&H0,&H0
,&H0,&H0,&H00A,&H0
NJ 4160 DATA &HAA00,&HAA0,&H0,&HAA0
0A,&H0,&HAA00,&HAA0,&H0
OL 4170 DATA &HAA00A,&H0,&HAA00,&H
A0,&H0,&HAA00A,&H0,&HAA00
NB 4180 DATA &HAA0,&H0,&HAA00A,&H0
,&HAA00,&HAA0,&H0,&HAA00A
EN 4190 DATA &H0,&HAA00,&H0,&H0,&H0
,&H0,&H0,&H0,&H2A00
HD 4200 DATA &HABAA,&H0,&HAA2A,&
HAB,&H0
PD 4210 REM CURSOR.BLU SHAPE S8
PH 4220 DATA &H20,&H14,&H1500,&H
5455,&H0,&H5515,&H54,&H0
BJ 4230 DATA &H0,&H0,&H0,&H0,&H5
0,&H0,&H5005,&H0
JB 4240 DATA &H500,&H50,&H0,&H50
05,&H0,&H500,&H50,&H0
MP 4250 DATA &H5005,&H0,&H500,&H
50,&H0,&H5005,&H0,&H500
WJ 4260 DATA &H50,&H0,&H5005,&H0
,&H500,&H50,&H0,&H5005
DB 4270 DATA &H0,&H500,&H0,&H0,&H0
,&H0,&H0,&H0,&H1500
KF 4280 DATA &H5455,&H0,&H5515,&
H54,&H0
MK 4290 REM PLUS.PUR SHAPE S9
KE 4300 DATA &H1C,&H6,&HA,&HA,&H
A,&HA,&HAA0AA,&HAA0AA
DN 4310 DATA &HAA0AA,&HAA0AA,&HA,&
HA,&HA,&HA,&H500
PB 4320 REM PLUS.BLU SHAPE S11
FJ 4330 DATA &H1C,&H6,&H5,&H5,&H
5,&H5,&H5055,&H5055
PK 4340 DATA &H5055,&H5055,&H5,&
H5,&H5,&H5,&H500

```

Program 7. Hex War For Amiga

Version by Philip Nelson, Assistant Editor

' Hex War for 512K Amiga

```

4
CLEAR ,250004
CLEAR ,655364
4
Restart:4
GOSUB Setup4
4
mainloop:4
GOSUB Reveille4
GOSUB DrawField4
GOSUB PlaceTroops4
GOSUB TakeTurn4
CLS:talk$="Thinking"4
LOCATE 12,17:PRINT talk$4
GOSUB talk4
GOSUB Battle4
GOSUB Resolve4
GOSUB Battle4
GOSUB Prisoners4
GOSUB Reinforcements4
GOSUB Outcome4
pn=1-pn:ft=0:pp=04

```

```

GOTO mainloop4
4
DrawField:4
CLS4
FOR r=11 TO 1 STEP -24
FOR c=12-r TO r+26 STEP 44
PUT (c*8,r*8),s104
NEXT c,r4
FOR r=13 TO 21 STEP 24
FOR c=r-10 TO 49-r STEP 44
PUT (c*8,r*8),s104
NEXT c,r4
FOR r=12 TO 1 STEP -24
FOR c=14-r TO r+28 STEP r+28-(14-r)-14
LOCATE r,c4
PRINT CHR$(32)4
NEXT c,r4
FOR r=13 TO 21 STEP 24
FOR c=r-11 TO 53-r STEP 53-r-(r-11)-14
LOCATE r,c4
PRINT CHR$(32)4
NEXT c,r4
FOR j=2 TO 23 STEP 214
LOCATE j,124
PRINT SPACE$(19)4
NEXT4
FOR j=1 TO 124
GOSUB 7104
NEXT4
j=14
IF gn=1 THEN GOSUB 718:j=2:GOSUB
7154
IF gn=2 THEN GOSUB 718:j=3:GOSUB
7154
LOCATE 1,14
WHILE INKEY$<>"":WEND4
RETURN4
710 k=cit(j,0)4
l=cit(j,1)4
x=(k-1)*2+194
y=(12-(k+1))*2+34
PUT (x*8+y*8+3),s2,PSET4
RETURN4
715 k=cit(j,0)4
l=cit(j,1)4
x=(k-1)*2+194
y=(12-(k+1))*2+34
PUT (x*8+y*8),s3,PSET4
RETURN4
718 k=cit(j,0)4
l=cit(j,1)4
x=(k-1)*2+194
y=(12-(k+1))*2+34
PUT (x*8+y*8),s4,PSET4
RETURN4
4
TakeTurn:4
IF nx(pn)<2 THEN RETURN4
ht=4:hb=4:GOSUB 10004
810 mv=0:ct=04
cb=0:pk=0:k=04
FOR j=1 TO nx(pn)-14
IF army(j,0,pn)>0 AND army(j,6,p
n)<1 THEN4
k=14
j=nx(pn)-14
END IF4
NEXT j4
IF k=0 THEN RETURN4
CheckIt:4
IF a$=CHR$(27) THEN a$="":RETURN
4
ReadMouse:4
IF MOUSE(0)<>2 THEN NoFlag4
' left button clicked twice4
WINDOW 4,"Speech",(65,70)-(225,1
10),16,14
IF TalkFlag=1 THEN4
talk$="Now I can talk."4
PRINT talk$4
TalkFlag=1-TalkFlag4
GOSUB talk4
GOTO ClearMouse4

```



```

END IF
IF TalkFlag=0 THEN
talk$="OK, I'll be quiet."
PRINT talk$
GOSUB talk
TalkFlag=1-TalkFlag
END IF
ClearMouse:
WHILE MOUSE(0)<>0:WEND
PRINT "Press button once"
PRINT "to continue..."
' wait for one click
WHILE MOUSE(0)<>1:WEND
' purge keyboard, too
WHILE INKEY$<>"":WEND
WINDOW CLOSE 4
NoFlag:
qv=0:a$=INKEY$:IF a$="" THEN Rea
dMouse:
IF UCASE$(a$)="Q" THEN
GetOut:
WINDOW CLOSE 3
SCREEN CLOSE 1
WINDOW 1,"Hex War",,31,-1
WINDOW OUTPUT 1
CLEAR ,25000
END
END IF
IF a$=CHR$(30) THEN qv=3:GOTO Co
deit
IF a$=CHR$(31) THEN qv=7:GOTO Co
deit
IF a$=CHR$(28) THEN qv=1:GOTO Co
deit
IF a$=CHR$(29) THEN qv=5
Codeit:
j=qv-128*(a$=" ")
IF j=0 THEN CheckIt
IF (j AND 128) THEN 1100
IF (j AND 1)=0 THEN CheckIt
j=(j-1)/2
IF j AND 1 THEN bl=hb+j-2:tl=ht
ELSE tl=ht+1-j:bl=hb
IF tl<0 OR tl>8 THEN CheckIt
IF bl<0 OR bl>8 THEN CheckIt
sl=tl+bl:IF sl<4 OR sl>12 THEN C
heckIt
hb=bl:ht=tl:GOSUB 1000
LOCATE 1,1
FOR z=1 TO 6
PRINT SPACE$(8)
NEXT
qn=map(ht,hb,0)
IF qn=0 THEN LOCATE 1,1:GOTO Che
ckIt ELSE q1=map(ht,hb,1)-1
LOCATE 1,1
PRINT USING "####";qn:PRINT"----
--"
FOR j=0 TO 3
PRINT USING "####";army(qn,j,q1)
NEXT
LOCATE 1,1
GOTO CheckIt
1000 sx=146+16*(ht-hb)
sy=210-16*(ht+hb)
IF map(ht,hb,2)=1 THEN
PUT (ox,oy),s8
PUT (sx,sy),s7
ox=sx:oy=sy
pp=1
RETURN
END IF
IF pp THEN
PUT (ox,oy),s7
PUT (sx,sy),s8
ox=sx:oy=sy
pp=0
RETURN
END IF
IF ft THEN
PUT (ox,oy),s8
PUT (sx,sy),s8
ox=sx:oy=sy

```

```

RETURN
END IF
PUT (sx,sy),s8
ox=sx:oy=sy:ft=1
RETURN
1100 IF pk=1 THEN 1200
IF map(ht,hb,1)<>pn+1 OR map(ht,
hb,0)=0 THEN 810
an=map(ht,hb,0)
IF army(an,6,pn)<>0 THEN 810
pk=1:ct=ht:cb=hb
cs=army(an,0,pn):SOUND 1100,10
talk$=STR$(cs)+CHR$(32)+"roahboh
ts.":GOSUB talk
GOTO CheckIt
1200 j=(ht=ct) AND (hb=cb)
IF j AND mv=0 THEN 810
IF j AND mv>0 THEN 1420
ax=army(map(ht,hb,0),0,pn)
IF (ax>me AND cs>me) OR (map(ht,
hb,1)-1=1-pn AND ax>0) THEN Chec
kIt
dt=ABS(ct-ht)
db=ABS(cb-hb)
tl=db+dt
IF NOT (tl=1 OR (ct+cb=ht+hb AND
dt=1)) THEN CheckIt
mg=map(ht,hb,0)
IF mg=0 THEN 1300
FOR j=0 TO 3
army(mg,j,pn)=army(mg,j,pn)+army
(an,j,pn)
army(an,j,pn)=0
NEXT
army(mg,6,pn)=1
map(ct,cb,0)=0
cs=army(mg,0,pn)
an=mg:mv=mm+1
GOTO 1300
1300 n8=map(ht,hb,1)-1
mv=mv+1
IF n8<>pn THEN mv=mv+1
map(ct,cb,0)=0
map(ht,hb,0)=an
map(ht,hb,1)=pn+1
army(an,4,pn)=ht
army(an,5,pn)=hb
IF mv>mm THEN army(an,6,pn)=1
k=0
FOR j=-1 TO 1 STEP 2
j1=ht+j:j2=hb+j:j3=hb-j
IF j1<0 OR j1>8 THEN 1340
IF map(j1,hb,0)>0 THEN
IF map(j1,hb,1)=2-pn THEN
k=1:j=1:GOTO 1360
END IF
END IF
1340 IF j2<0 OR j2>8 THEN 1350
IF map(ht,j2,0)>0 THEN
IF map(ht,j2,1)=2-pn THEN
k=1:j=1:GOTO 1360
END IF
END IF
1350 IF j3<0 OR j3>8 OR j1<0 OR
j1>8 THEN 1360
IF map(j1,j3,0)>0 THEN
IF map(j1,j3,1)=2-pn THEN k=1:j=
1
END IF
1360 NEXT j
IF k=1 THEN army(an,6,pn)=1:mv=mm
+1
1380 a=pn:j=ct
k=cb:c=0:d=0
GOSUB 1830
j=ht:k=hb
c=cs:d=army(an,6,pn)
GOSUB 1830
ct=ht:cb=hb
IF mv<mm THEN CheckIt
1420 army(an,6,pn)=1
j=ht:k=hb

```

```

c=cs:d=1
GOSUB 1830
GOTO 810
1500 RESTORE Strengths
FOR j=0 TO 14
nx(j)=5
FOR k=1 TO 4
READ a,b,c
army(k,0,j)=a
army(k,4,j)=b
army(k,5,j)=c
map(b,c,0)=k
map(b,c,1)=j+1
NEXT k,j
Strengths:
DATA 64,2,8,64,3,7,64,5,6,64,6,6
DATA 64,2,2,64,3,2,64,5,1,64,6,0
FOR j=0 TO 14
FOR k=0 TO 20
a=INT(RND(1)*k*3)
FOR l=1 TO 5
a=a+INT(RND(1)*21-8)
NEXT l
IF a<16 THEN a=0 ELSE a=(a+k*8)
AND 254
fq(k,j)=a
NEXT k,j
RETURN
PlaceTroops:
FOR j=0 TO 8
FOR k=0 TO 8
a=map(j,k,1)
IF a THEN a=1:GOSUB 1800
NEXT k,j
FOR a=0 TO 1
e=13+a*12:f=a*22
dx=2-4*a:d=0
FOR j=0 TO 8
c=fq(j,a):GOSUB 1840
e=e+dx*2
IF j>3 THEN f=f+dx:e=e-dx
NEXT j,a
IF pn THEN
PUT (280,160),s5,PSET
ELSE
PUT (280,160),s6,PSET
END IF
RETURN
1800 b=map(j,k,0)
c=army(b,0,a)
d=army(b,6,a)
1830 e=(j-k+10)*2-1
f=(13-j-k)*2+1
1840 IF a THEN
PUT (e*8+1,f*8),s5,PSET
LOCATE f+a+1,e+1
PRINT SPACE$(2);
GOTO 1850
END IF
PUT (e*8+1,(f+1)*8),s6,PSET
LOCATE f+a+1,e+1
PRINT SPACE$(2);
1850 IF c=0 THEN RETURN
LOCATE f+a+1,e+1
PRINT RIGHT$("0"+HEX$(c),2);
IF d AND a=0 THEN
LOCATE f+2,e+1
PRINT SPACE$(2);
PUT (e*8+1,(f+1)*8+1),s11
RETURN
END IF
IF d AND a=1 THEN
LOCATE f+1,e+1
PRINT SPACE$(2);
PUT (e*8+1,f*8+1),s9
END IF
RETURN
Reveille:

```



```

sw=0:e=nx(pn)-1
IF e<1 THEN RETURN
FOR j=1 TO e-1
IF army(j,0,pn)>=1 THEN 1970
t=army(j,4,pn)
b=army(j,5,pn)
IF map(t,b,0)=j THEN map(t,b,0)=
0
FOR k=j TO e
FOR l=0 TO 6
army(k,1,pn)=army(k+1,1,pn)
army(k+1,1,pn)=0
NEXT l
t=army(k,4,pn)
b=army(k,5,pn)
map(t,b,0)=k
NEXT k
nx(pn)=nx(pn)-1
j=e: sw=1
1970 NEXT j
IF sw THEN Reveille
FOR j=1 TO e
army(j,0,pn)=army(j,0,pn)+army(j
,2,pn)
army(j,2,pn)=army(j,3,pn)
army(j,3,pn)=0
army(j,6,pn)=0
NEXT j
k=nx(1-pn)
FOR j=1 TO k
army(j,6,1-pn)=0
NEXT
GOSUB 2400
2050 IF bp>0 THEN
FOR j=0 TO 1
FOR k=1 TO bp
a=BTL(k,j,0)
army(a,6,j)=army(a,6,j)+1
NEXT
NEXT
END IF
RETURN
Battle:
GOSUB 2400
a=nx(0)
IF nx(1)>a THEN a=nx(1)
FOR j=0 TO 1
FOR k=1 TO a
army(k,6,j)=0
NEXT k
GOSUB 2050
RETURN
Prisoners:
FOR j=0 TO 1
a=1-j
b=nx(j)-1
FOR k=1 TO b
IF army(k,0,j)>=1 THEN 2280
fq(2,a)=fq(2,a)+army(k,2,j)+army
(k,3,j)
IF fq(2,a)>255 THEN
c=fq(2,a)-255
fq(3,a)=fq(3,a)+c
fq(2,a)=255
END IF
fq(6,a)=fq(6,a)+army(k,1,j)
IF fq(6,a)>255 THEN
c=fq(6,a)-255
fq(7,a)=fq(7,a)+c
fq(6,a)=255
END IF
IF map(army(k,4,j),army(k,5,j),0
)=k AND map(army(k,4,j),army(k,5
,j),1)=j+1 THEN
map(army(k,4,j),army(k,5,j),0)=0
END IF
FOR l=0 TO 6
army(k,1,j)=0
NEXT
2280 IF army(k,6,j)>=1 THEN 2320
fq(4,j)=fq(4,j)+army(k,1,j)

```

```

army(k,1,j)=0
IF fq(4,j)>255 THEN
c=fq(4,j)-255
fq(5,j)=fq(5,j)+c
fq(4,j)=255
END IF
2320 NEXT k,j
RETURN
2400 bp=0
FOR j=0 TO 8
j1=(j-4)*(4-j)
j2=8-(j+4)*(4-j)
FOR k=j1 TO j2
a=map(j,k,0)
r=map(j,k,1)
IF a=0 OR r=0 THEN 2490
IF army(a,0,r-1)<1 THEN 2490
t=j+1
b=k: GOSUB 2500
b=b-1: GOSUB 2500
t=t-1: GOSUB 2500
2490 NEXT k,j
RETURN
2500 IF t<0 OR b<0 OR t>8 OR b>8
THEN RETURN
pa=map(t,b,0)
IF pa=0 THEN RETURN
IF map(t,b,1)=r THEN RETURN
IF army(pa,0,2-r)<1 THEN RETURN
bp=bp+1
BTL(bp,r-1,0)=a
BTL(bp,2-r,0)=pa
RETURN
Resolve:
IF bp=0 THEN RETURN
FOR j=1 TO bp
FOR k=0 TO 1
a=1-k
an=BTL(j,k,0)
ax=army(an,0,k)
ht=army(an,6,k)
ct=INT(ax/ht)+1
BTL(j,a,1)=INT(ct*ka+1)
BTL(j,a,2)=INT(ct*kb+1)
BTL(j,a,3)=INT(ct*kc+1)
NEXT k,j
FOR j=1 TO bp
j0=BTL(j,0,0)
j1=BTL(j,1,0)
GOSUB 3100
army(j0,0,0)=army(j0,0,0)-a*BTL(
j,0,1)
army(j1,0,1)=army(j1,0,1)-b*BTL(
j,1,1)
GOSUB 3100
c=a*BTL(j,0,2)
army(j0,0,0)=army(j0,0,0)-c
army(j0,1,0)=army(j0,1,0)+c
c=b*BTL(j,1,2)
army(j1,0,1)=army(j1,0,1)-c
army(j1,1,1)=army(j1,1,1)+c
GOSUB 3100
c=a*BTL(j,0,3)
army(j0,0,0)=army(j0,0,0)-c
army(j0,3,0)=army(j0,3,0)+c
c=b*BTL(j,1,3)
army(j1,0,1)=army(j1,0,1)-c
army(j1,3,1)=army(j1,3,1)+c
NEXT j
RETURN
Reinforcements:
a=1-pn:b=0
FOR j=0 TO 8:FOR k=0 TO 8
IF map(j,k,1)=pn+1 THEN b=b+1
NEXT k,j
fq(1,pn)=fq(1,pn)+b
IF fq(1,pn)>255 THEN
b=fq(1,pn)-255
fq(2,pn)=fq(2,pn)+b
fq(1,pn)=255
END IF

```

```

t=4:b=pn*8
IF map(t,b,0)<>0 THEN RETURN
IF map(t,b,1)=pn+1 THEN
fq(0,a)=0:fq(1,a)=0
GOTO 3060
END IF
j=nx(a)
IF j>31 THEN RETURN
j1=fq(0,a)
IF j1<1 THEN 3060
nx(a)=nx(a)+1
map(t,b,0)=j
map(t,b,1)=a+1
army(j,0,a)=j1
FOR k=1 TO 3
army(j,k,a)=0
NEXT k
army(j,4,a)=t
army(j,5,a)=b
3060 FOR k=0 TO 19
fq(k,a)=fq(k+1,a)
NEXT k
fq(20,a)=0
RETURN
3100 a=0:FOR m=1 TO 6
IF RND(1)<.5 THEN a=a+1
NEXT m:b=6-a
RETURN
3200 talk$="press 1 through 5 to
choose seenaireeo."
GOSUB talk
WINDOW 4,"Scenario: Press 1-5",
(65,70)-(255,120),16,1
PRINT "1> Capture capital/far"
PRINT "2> Capture capital/near"
PRINT "3> Occupy 8/12 cities"
PRINT "4> Control 6/12 cities"
PRINT "5> Occupy 40/61 hexes"
GrabKey:
a$=INKEY$:IF a$="" THEN GrabKey
gn=VAL(a$)
IF gn<1 OR gn>5 THEN GrabKey
WINDOW CLOSE 4
talk$="seenaireeo"+STR$(gn)+CHR$
(46):GOSUB talk
RETURN
Outcome:
a=0:ON gn GOSUB 3430,3450,3480,3
490,3580
IF a=0 THEN RETURN
en=c$:ea=a
GOSUB DrawField
GOSUB PlaceTroops
a=ea
WINDOW 4,"Outcome", (25,70)-(300,
120),16,1
PRINT "Player "a" wins"
MaybeOut:
PRINT c$
PRINT "Press Q to quit, RETURN t
o play."
a$=""
WHILE a$=""
a$=INKEY$
WEND
WINDOW CLOSE 4
IF UCASE$(a$)="Q" THEN GetOut
WINDOW CLOSE 2:WINDOW CLOSE 1
CLEAR ,25000
RUN
3430 IF map(cit(2,0),cit(2,1),1)
=1 THEN
a=2
c$="Red captured the capital"
GOSUB Announce
RETURN
END IF
GOTO 3460
3450 IF map(cit(3,0),cit(3,1),1)
=1 THEN

```



```

a=24
c$="Red captured the capital"4
GOSUB Announce4
RETURN4
END IF4
4
3460 IF map(cit(1,0),cit(1,1),1)
=2 THEN4
a=14
c$="Yellow captured the capital"
4
GOSUB Announce4
RETURN4
END IF4
RETURN4
4
3480 l=8:GOTO 35004
3490 l=64
3500 c(1)=0:c(2)=04
FOR j=1 TO 12:t=cit(j,0):b=cit(j
,l)4
r=map(t,b,1):c(r)=c(r)+14
IF gn=4 THEN4
an=map(t,b,0)4
IF r>0 THEN IF an=0 OR army(an,6
,r-1)>0 THEN c(r)=c(r)-14
END IF4
NEXT j4
IF c(1)>=1 THEN4
a=24
c$="Red captured"+STR$(c(1))+ " c
ities"4
GOSUB Announce4
RETURN4
END IF4
IF c(2)>=1 THEN4
a=14
c$="Yellow captured"+STR$(c(2))+
" cities"4
GOSUB Announce4
END IF4
RETURN4
4
3580 c(1)=0:c(2)=04
FOR j=0 TO 8:FOR k=0 TO 84
r=map(j,k,1):c(r)=c(r)+14
NEXT k,j4
IF c(1)>=40 THEN4
a=24
c$="Red occupies"+STR$(c(1))+ " h
exes"4
talk$="REHD AA4KYUWPAYZ":SAY tal
k$4
talk$=STR$(c(1))+ "hexes"4
GOSUB talk:RETURN4
END IF4
IF c(2)>=40 THEN4
a=14
c$="Yellow occupies"+STR$(c(2))+
" hexes"4
talk$="YEHLOH AA4KYUWPAYZ":SAY t
alk$4
talk$=STR$(c(2))+ "hexes"4
GOSUB talk:RETURN4
END IF4
RETURN4
4
Setup:4
DEFINT s4
SCREEN 1,320,200,2,14
' open window 3 with no 4
' gadgets or title bar4
WINDOW 1,"", (0,0)-(311,25),16,14
WINDOW 3,"", (0,0)-(311,185),16,1
4
WINDOW OUTPUT 34
PALETTE 0,0,0,04
PALETTE 1,.5,1,14
PALETTE 2,1,0,04
PALETTE 3,1,1,.14
WIDTH 404
CLS4
DIM Voice$(8)4
RESTORE VoiceData4
FOR j=0 TO 84
READ Voice$(j)4
NEXT4
RESTORE4
' speech will be synchronous4
VoiceData:4
DATA 110,0,170,0,22200,64,10,1,0
4
talk$="Welcome to Hex War."4
LOCATE 13,114
PRINT talk$4
GOSUB talk4
Temp$="Click button twice to tur
n"4
LOCATE 15,8:PRINT Temp$4
Demp$=" speech off or on during
game."4
LOCATE 16,6:PRINT Demp$4
talk$=Temp$+Demp$4
GOSUB talk4
' hex shape4
LINE (0,0)-(2,0):LINE (13,0)-(15
,0)4
LINE (0,1)-(3,1):LINE (12,1)-(15
,1)4
LINE (3,2)-(12,2):LINE (4,3)-(11
,3)4
FOR j=4 TO 114
LINE (6,j)-(9,j)4
NEXT4
LINE (4,12)-(11,12):LINE (3,13)-
(12,13)4
LINE (0,14)-(3,14):LINE (12,14)-
(15,14)4
LINE (0,15)-(2,15):LINE (13,15)-
(15,15)4
DIM s10(225)4
GET (0,0)-(15,15),s104
PUT (0,0),s104
' cursor shape4
FOR k=0 TO 14
GOSUB Bracket4
PAINT (32,42),2+k,14
LINE (30,35)-(48,44),0,bf4
LINE (32,32)-(47,47),0,bf4
GOSUB Bracket4
IF k=0 THEN4
DIM s8(400)4
GET (26,26)-(53,49),s84
PUT (26,26),s84
END IF4
IF k=1 THEN4
DIM s7(400)4
GET (26,26)-(53,49),s74
PUT (26,26),s74
END IF4
NEXT k4
GOTO Blip4
Bracket:4
PUT (16,32),s104
PUT (32,48),s104
PUT (48,32),s104
PUT (32,16),s104
RETURN4
Blip:4
' city shape4
FOR j=0 TO 14
LINE (2,j)-(7,j)4
LINE (2,j+8)-(7,j+8)4
LINE (j,2)-(j,7)4
LINE (j+8,2)-(j+8,7)4
NEXT4
PSET (1,1):PSET (8,1)4
PSET (1,8):PSET (8,8)4
DIM s2(100)4
GET (0,0)-(9,9),s24
PUT (0,0),s24
' capital shape4
FOR k=0 TO 14
FOR j=0 TO 34
LINE (4,j)-(11,j),2+k4
LINE (4,j+12)-(11,j+12),2+k4
LINE (1,4+j)-(14,4+j),2+k4
LINE (1,8+j)-(14,8+j),2+k4
NEXT4
LINE (3,6)-(5,9),0,bf4
LINE (6,3)-(9,5),0,bf4
LINE (6,10)-(9,12),0,bf4
LINE (10,6)-(12,9),0,bf4
IF k=1 THEN4
DIM s3(225)4
GET (0,0)-(15,15),s34
PUT (0,0),s34
END IF4
NEXT k4
' army shape4
FOR j=0 TO 144
LINE (7,0)-(j,7),34
NEXT4
FOR j=4 TO 104
LINE (7,2)-(j,5),04
NEXT4
DIM s5(64)4
GET (0,0)-(14,7),s54
PUT (0,0),s54
' other army shape4
FOR j=0 TO 144
LINE (7,7)-(j,0),24
NEXT4
FOR j=4 TO 104
LINE (7,5)-(j,2),04
NEXT4
DIM s6(64)4
GET (0,0)-(14,7),s64
PUT (0,0),s64
' crosses
FOR k=0 TO 14
FOR j=0 TO 14
LINE (0,2+j)-(13,2+j),2+k4
LINE (10+j,0)-(10+j,5),2+k4
LINE (2+j,0)-(2+j,5),2+k4
NEXT4
FOR j=0 TO 14
LINE (6+j,0)-(6+j,5),04
NEXT4
IF k=0 THEN4
DIM s11(150)4
GET (0,0)-(13,13),s114
PUT (0,0),s114
END IF4
IF k=1 THEN4
DIM s9(150)4
GET (0,0)-(13,13),s94
PUT (0,0),s94
END IF4
NEXT k4
DIM army(31,6,1),BTL(64,1,3)4
DIM map(9,9,2),fq(20,1),nx(1),c(
2)4
cn=12:DIM cit(cn,1)4
RANDOMIZE TIMER4
4
pn=1:me=314
mm=3 ' Maximum number of moves4
ka=1/48:kb=1/48:kc=1/324
RESTORE Whatsit4
FOR j=1 TO cn4
FOR k=0 TO 14
READ cit(j,k)4
NEXT4
map(cit(j,0),cit(j,1),2)=14
NEXT4
Whatsit:4
DATA 8,4,0,4,8,0,0,8,4,0,4,84
DATA 5,5,3,3,6,3,2,5,5,2,3,64
CLS4
GOSUB 32004
CLS4
GOSUB 15004
RETURN4
4
Announce:4
talk$=c$4
4
talk:4
IF TalkFlag=0 THEN SAY TRANSLATE
$(talk$),Voice$4
RETURN4
4

```


Leader Board For The 64

David Florance
Programming Assistant

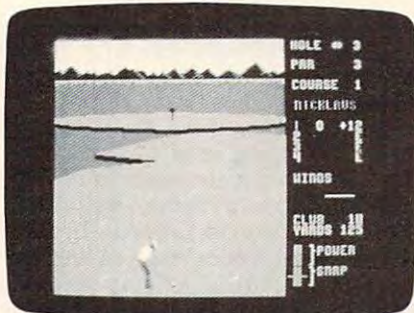
Requirements: Commodore 64 (or Commodore 128 in 64 mode) with a disk drive and a joystick. Versions for the Amiga and Atari ST are planned.

The spring and summer months, with their profusion of golfing events, couldn't be a better time for Access Software to have released its new *Leader Board* professional golf simulator. Continuing in the tradition of such earlier popular releases as *Beach-Head* and *Raid Over Moscow*, Access has fashioned a stunning piece of software in this new golfing game.

All who have tested *Leader Board* agree that it has excellent sound and graphics and is a lot of fun to play. *Leader Board* is easy to use, too. Although you probably won't shoot under-par scores during your first 18 holes, we've yet to see someone play the game and not like it. One person trying *Leader Board* for the first time scored a 52 on one hole (for you golf novices, that's not very good), and still said he enjoyed the game. Another player, professing to like neither computers nor golf, is considering buying a computer just to play *Leader Board*.

Bruce and Roger Carver, authors of the game, have done an exceptional job on everything from the golfer's swing to the action of the joystick. The program lets you hook, slice, cut, plug, top, and drub—just as in real life. You can even learn to hit the ball straight—if you concentrate.

Leader Board offers three levels of play: novice, amateur, and professional. Start with the novice level to get some practice. You can even move to the driving range for additional practice on your strokes. The program lets you play anywhere from 18 to 72 holes, and there are four courses from which to choose—each with a distinct personality and level of difficulty. Even the



Teeing off on a typical hole in Leader Board, an exceptional golf simulator for the Commodore 64.

wind is a big factor on the professional level of *Leader Board*.

Good Whooshes And Plops

From one to four players can take part, and scoring is automatic. The sound effects—from the whoosh of the stroke to the plop of the ball landing in a water hazard—are excellent throughout. Even the sound of the ball dropping into the cup is realistic. The movements of the golfer and the ball in flight (and bouncing on the fairway or green) are superb.

Just as in a real game of golf, you'll need some time to get your strokes down. You control the power of your swing and the direction of the ball by pressing the joystick button and moving the stick forward or backward at the right moments.

After playing hundreds of holes, I've concluded that the most important factor in making good scores is selecting the right clubs. *Leader Board's* manual offers course cards with detailed yardage indicators as well as a chart with normal club distances. These are invaluable. Access Software is also selling additional tournament disks, with four different courses on each disk, for \$19.95 each.

The *Leader Board* disk is not copy-protected, so you can make backups for safekeeping. None of the disks work, however, unless a security key is plugged into the computer's cassette port.

I've been a golfer for about 15 years. Maybe it's a coincidence, but

when I went to my local course after playing *Leader Board* for several weeks, I had a great round. Who knows? Maybe *Leader Board* is even improving my game.

Leader Board
Access Software, Inc.
2561 South 1560 West
Woods Cross, UT 84087
\$39.95

SunDog: Frozen Legacy For Atari ST

David Florance
Programming Assistant

Requirements: Atari ST-series computer with a color monitor; or an Apple II-series computer with at least 64K RAM and a color monitor. The ST version was reviewed.

Certainly one of the most exciting aspects of the future is space exploration. How will it be out there? What will we find, and how will we learn to adapt and go about our everyday existence? Will people carry the same instincts, societal norms, beliefs, and habits into the dark reaches of space? Whatever happens, it's a sure bet that if life's at all similar to *SunDog: Frozen Legacy*, we'll still have to know what's a good deal and what's not, when to beg, when to borrow, and when to...well, rethink our priorities.

SunDog, from Oasis/FTL games, is a first-rate graphics adventure with enough complexity for the seasoned player but simple enough for a beginner to enjoy. First marketed for the Apple II computers, the new Atari ST version features stunning graphics and easy mouse-driven controls.

You start this adventure with a tremendous inheritance left to you by an ambitious uncle who had designs on building a religious colony. Your task, a large one, is to fill his shoes by completing his dream. It's not easy. There's

more than one obstacle in your path. You don't know where the colony was planned, how to pilot a freighter (the *SunDog*), or how to spend the money he left you. Be careful—chances are you'll be mugged, swindled, and/or lost in a vast mountainous continent before you know it. You can even lose your starship and your inheritance if you don't pay attention to business.

Your first task is to find the colony (named Banville). There's an immense amount of ground to cover because *SunDog: Frozen Legacy* encompasses 50 cities on 18 different inhabited planets in 12 star systems.

Next, you'll have to locate and buy all the goods necessary to complete the colony. Although you start off with enough money to purchase the goods, unless you are very adroit, lucky, or both, you'll make some financial mistakes that may exhaust or at least severely deplete your bank account. Don't despair. Just start over and be more careful next time. Remember, no one promised that life in the far reaches of space would be a rose garden.

Beware Of Beggars

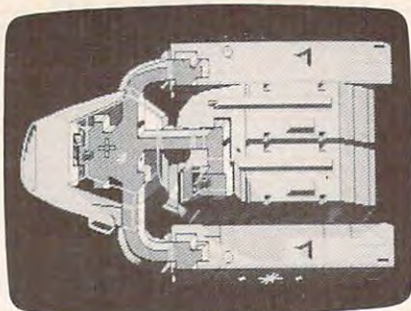
Interaction is an important part of *SunDog: Frozen Legacy*. The shopkeepers and store managers talk to you, and you're expected to respond. Your range of possible responses is limited, though this might be fortunate. If you utter the things that run across your mind while you're trying to finish the mission, you'd probably be hunted down by the galactic security force and executed on sight. So don't be too abrasive.

On the other hand, when a beggar asks you for money, don't give him your last piece of cash. You may need it to transport hastily out of the city. Some of those beggars are ruffians as well, so be prepared to defend yourself. In other words, even in this futuristic setting, the general populace seems still to agree with P.T. Barnum's dictum that a sucker's born every minute. And for the most part, you are considered the latest.

Other characters to watch out for are the many entrepreneurs wandering around or hanging out in the eating establishments. They're more than willing to be your pal. You'll have to be clever to sort out the good and bad deals. These gents can help you as well as harm you.

Popsicle People

To finish the adventure, you have to locate the colonists. They aren't capable of finding you because they're cryogenically frozen. This is when your piloting skills aboard the *SunDog* are put to the test. Setting the course is perhaps the



A top view of the *SunDog*, a space freighter in which you search for frozen colonists and a lost colony (Atari ST version).

most challenging level of the adventure. There are many ways to be successful, and just as many ways to fail. For instance, you can save time by traveling at warp speeds, but you'll spend more fuel, and trying to reach warp can be dangerous. To be safe, be sure to check for engine damage often. When the *SunDog* is not operating at peak efficiency, you lose time and effort. Buying replacement parts is not difficult, and if you shop around you may

be able to find some bargains.

The Atari ST gets a chance to show off with *SunDog: Frozen Legacy*. Stunning visual effects abound, and each level has graphic screens that will amaze you. The detail is enormous. If you've been wondering what a graphics adventure game is like on the new generation of personal computers, *SunDog: Frozen Legacy* is a must. It's a whole different world.

A helpful hint from this Star Freighter Captain. Don't let your eyes fool you. You must still eat and sleep to survive and complete the mission. So take time out to catch a few winks and grab a bite to eat. You'll need all your strength and attention to conquer the challenges of *SunDog*.

SunDog: Frozen Legacy
(Atari ST version) Oasis/FTL Games
P.O. Box 112489
San Diego, CA 92111
(Apple II version) Accolade
Entertainment Software
20863 Stevens Creek Blvd.
Cupertino, CA 95014
\$39.95 each

The Goonies And Zorro

Karen McCullough

Requirements: The Goonies—an Apple II-series computer with at least 48K RAM and a disk drive; Commodore 64 or 128 (in 64 mode) with a 1541 disk drive; or an Atari 400/800/XL/XE with at least 48K RAM and a disk drive. Zorro—same requirements, except Apple II-series computers must have at least 64K RAM. Color monitor optional but recommended. Joystick required. The Apple versions were reviewed.

Are you tired of shooting aliens and centipedes in arcade-style computer games? Bored with piloting helicopters and drilling holes in brick walls? Does it seem like you've done it all—dodged the best of them, shot the worst—so that it's all a bit of a drag now? Don't give up yet. Datasoft/IntelliCreations has some new challenges for you: games with a smooth blend of arcade action and adventure-like puzzle-solving.

Help the Goonies (that famous band of adventurers from the movie of the same name) negotiate an underground labyrinth to find pirate treasure and save their parents' home from foreclosure. Or perhaps you'd rather be Zorro, using cunning and a sharp sword to rescue a lovely princess from Sergeant Garcia.

Whichever fantasy you choose, you'll need the standard arcade equipment: a keen eye and quick reflexes. But in these games, there's more to negotiating an underground maze than jumping over cannonballs and dodging bats. What do you do about steampipes that block your path? And how can you scale new heights without a ladder? If you consider it unreasonable to have to think about a problem rather than shoot your way out, you'd better consider another game.

Still interested? Good. These two games have a lot to recommend them: excellent graphics and animation; smooth, fast screen changes; and accurate control. At the start of each session with the Apple version, the program asks you to calibrate your joystick by moving the stick to the right-, left-, top-, and bottommost positions. Thereafter, the game adjusts itself to your stick settings, resulting in control as tight and as precise as found in most coin-operated videogames. (Joystick calibration isn't necessary with the Commodore and Atari versions.)

Inventive Puzzles

One of the best features of these games is their interesting and inventive puzzles. There's a lot happening on each screen, and it may take some time to figure out what it all means. In *The*

ATARI 130XE

ATARI 130XE Super Computer Package
130XE Computer 1027 Printer
1050 Disk Drive Atariwriter +
Call for individual & super package price

ATARI PRINTER INTERFACES

Uprint A 54.95 MPP 1150 54.95

Supra 1000E Modem 44.95

Atari XM-301 Modem 44.95

ATARI 130-XE SUPER PRINTER PACKAGES

NX-10 Printer & U-Print A 319

Panasonic 1091 & U-Print A 299

Super Printer Packages have no extra shipping charges or credit card surcharges when shipped in Continental USA

ATARI 130XE SOFTWARE

BRODERBUND 28.95

Print Shop 20.95

Karateka 20.95

Print Shop 20.95

Graph I, II, or III 19.95

Print Shop Comp 27.95

INFOCOM 27.95

See Commodore 64 section for items and prices

ELECTRONIC ARTS 24.95

Archon II 19.95

Archon 19.95

Seven Cit. of Gold 24.95

Skyfox 24.95

Pinball Const. 19.95

One on One 24.95

Super Boulder Dash 19.95

Chessmaster 2000 27.95

Racing Destruction 24.95

MICROPROSE 23.95

Silent Service 23.95

Gunsight 23.95

Acrojet 23.95

F-15 Strike Eagle 23.95

Kennedy Approach 23.95

Conflict/Vietnam 27.95

Synfile 32.95

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Mind Pursuit 23.95

Never-Ending Story 23.95

Foobitzky 27.95

Synfile 32.95

Typesetter 27.95

Ultima III 37.95

Ultima IV 41.95

Hacker 19.95

Beachhead II 27.95

Raid Over Moscow 27.95

Fight Night 23.95

Hardball 23.95

Flight Simulator II 34.95

Letter Perfect 39.95

Data Perfect 39.95

Alternate Reality 27.95

Fleet System II 49.95

Page Designer 23.95

Megafont II 19.95

Rubber Stamp 23.95

Sargon III 34.95

Halley Project 23.95

Synfile 32.95

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Zoomracks 49.95

Easy Draw 99.95

Mindshadow 34.95

Hippopotamus Call

Supra Hard Drive Call

Supra 1200 ST Modem Call

QMI 1200 ST Modem Call

PC Board Designer Call

Infocom See IBM

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Zoomracks 49.95

Easy Draw 99.95

Mindshadow 34.95

Hippopotamus Call

Supra Hard Drive Call

Supra 1200 ST Modem Call

QMI 1200 ST Modem Call

PC Board Designer Call

Infocom See IBM

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Zoomracks 49.95

Easy Draw 99.95

Mindshadow 34.95

Hippopotamus Call

Supra Hard Drive Call

Supra 1200 ST Modem Call

QMI 1200 ST Modem Call

PC Board Designer Call

Infocom See IBM

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Zoomracks 49.95

Easy Draw 99.95

Mindshadow 34.95

Hippopotamus Call

Supra Hard Drive Call

Supra 1200 ST Modem Call

QMI 1200 ST Modem Call

PC Board Designer Call

Infocom See IBM

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Zoomracks 49.95

Easy Draw 99.95

Mindshadow 34.95

Hippopotamus Call

Supra Hard Drive Call

Supra 1200 ST Modem Call

QMI 1200 ST Modem Call

PC Board Designer Call

Infocom See IBM

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Zoomracks 49.95

Easy Draw 99.95

Mindshadow 34.95

Hippopotamus Call

Supra Hard Drive Call

Supra 1200 ST Modem Call

QMI 1200 ST Modem Call

PC Board Designer Call

Infocom See IBM

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Zoomracks 49.95

Easy Draw 99.95

Mindshadow 34.95

Hippopotamus Call

Supra Hard Drive Call

Supra 1200 ST Modem Call

QMI 1200 ST Modem Call

PC Board Designer Call

Infocom See IBM

See Commodore 64 section for items and prices

MISCELLANEOUS 130XE 23.95

Amazon 34.95

DOS Shell 34.95

Kissed 34.95

9 Princess/Amber 27.95

Hacker 29.95

Dragonworld 34.95

Treasure Island 27.95

Wizard of Oz 27.95

Transylvania 27.95

Borrows Time 34.95

Mid-Term 34.95

Regent Spell 34.95

Regent Base Call

Goldrunner 27.95

Time Bandit 27.95

Goonies, the solution to a problem often requires getting the two characters on the screen to cooperate. In *Zorro*, you may need a special object, lots of curiosity, or just a bit of luck to solve a puzzle.

The two games are similar in concept, but not identical in execution. *The Goonies* has a stronger arcade action feel, as befits its descent from that exciting (if not very memorable) movie. *Zorro* plays more like a graphic adventure. There are objects to collect (some may have magical properties), a town to map, secret passages to find, and visual puzzles to solve. *The Goonies* has a hint sheet to help you figure out solutions; in *Zorro*, you're on your own.

Good as they are, though, neither game is perfect. *Zorro* could benefit from a hint sheet of its own—even with a good-quality color monitor it's difficult to tell what some of the objects are. One screen has something that looks like (but surely isn't?) a Coca-Cola bottle. A lantern, perhaps? A club? We aren't sure.

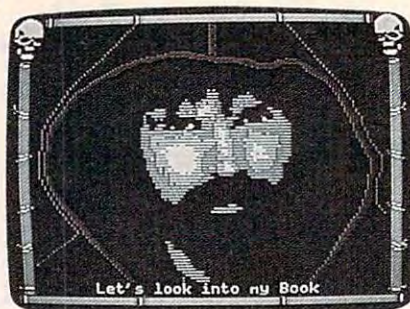
The Goonies needs a way to allow a player to practice on upper-level screens without going through all the lower ones. It takes some 20 minutes of playing time to get to the last levels; you arrive with only one or two lives left and promptly get zapped. That's acceptable at an arcade where the real

object is to entice you to spend another quarter, but in a home computer game, it can be frustrating. It would be nice to have an option similar to the one in *Lode Runner* which allows you to play on any level you wish, but prevents you from setting an official high score without progressing through the levels in proper order. Of course, high scores aren't much of a consideration in *The Goonies* or *Zorro*, since neither game saves these scores—another minor flaw.

One final quibble concerns the lack of an option for keyboard control, particularly in the Apple version. Many a white-collar computer runs games during lunch hour, but wouldn't dare be caught with anything so unprofessional as a joystick hanging out of its side. Most Apple games have a keyboard option for this reason.

Overall, however, *The Goonies* and *Zorro* are attractive games—fun, interesting, and entertaining. Due to the level of difficulty, they're not appropriate for children under ten, but older kids and adults will have a good time with them.

Zorro
The Goonies
 Datasoft/IntelliCreations, Inc.
 19808 Nordhoff Place
 Chatsworth, CA 91311
 Commodore and Atari versions \$29.95
 Apple versions \$39.95



A typically ominous screen from *Moebius: The Orb of Celestial Harmony*.

swing your sword to cut vegetation that blocks your way, use an item in your inventory, throw a shuriken, and much more.

The game is made even easier to play by the use of windowing. Windows often pop up on the main screen to offer various options. For example, pressing the C key to communicate opens a window and gives you the choice of asking a character for help, to follow you, to stay and wait for you, or to go away.

Much thought is required to play *Moebius* well. The mystery and intrigue of the Orient permeate the game, and virtuous behavior is often rewarded. You must think of others before yourself to be successful and preserve the purity of your Karma (very important).

Furthermore, strategy, planning, and quick thinking are a must. Poor villagers are afraid of men carrying swords and are averse to helping them; yet, hungry tigers cannot be fought off with your bare hands. It's up to you to decide when to arm yourself and when to trust in your karate skills.

Tiger Teeth And Panda Hair

The realm of magic is not ignored, either. In *Moebius*, however, magic works a little differently than in dungeon adventure games. Magic requires a strong mind, so you must fast and chant special mantras to activate such spells as Speak with the Dead, Waterwalk, or Cure Sickness. Likewise, your mind must be clear to divine the nature of artifacts. When this magic is combined with another component (tiger teeth, beetle pincers, condor feathers, etc.), you can teleport, use ventriloquism, cause paralysis, and invoke many other charms. Incidentally, the magical components must be found and then either purchased or earned. To get panda hair, for example, you must first trap the bear.

Moebius embodies its own unique playing style and feel, and it gives the player an unmistakable sense of the Far

Moebius: The Orb Of Celestial Harmony For Apple

James V. Trunzo

Requirements: Apple II-series computer with at least 64K RAM and a disk drive. Commodore 64/128 version scheduled for release by July.

Fresh on the heels of the celebrated release of *Ultima IV*, Origin Systems has produced yet another program worth raving about. *Moebius: The Orb of Celestial Harmony* capitalizes on the current popularity of ninjas by casting the player as a youthful martial arts disciple on a trip through an oriental world full of danger and excitement.

Moebius combines all the familiar elements of a computer role-playing game with an arcade-style combat system that's both challenging and functional. That is, the arcade-style combat wasn't included merely to show off the program's superior graphics. Your ninja disciple must become proficient with both a sword and his bare hands to defeat enemies and gain experience points that heighten various attributes.

The theme of *Moebius* is simple. After learning all he could from his ninja masters, a wayward disciple named Kaimen stole the Orb of Celestial Harmony. This act has brought much suffering to the land of Khantun. Kaimen has set himself up as supreme warlord and is conducting a reign of terror. He has imprisoned the Holy Ones, replacing them with his own evil monks. Monsters roam the land and infest the waters of Khantun. A savior is needed—a warrior who has devoted his life to Moebius and who has trained with the Sword Master, the Martial Arts Master, and the Zen Master (for all is not won through force of arms). You are that savior, the ninja warrior.

Protect Your Karma

Moebius employs a number of single-keystroke commands, much like other Origin games such as *Ultima* and its sequels. Your commands are varied: You can communicate with the many characters you encounter during your travels (even the skeletons of the victims of Kaimen—if you know magic),

HOTWARE: Software Best Sellers

| | | | | | Systems | | | | |
|------------------------|------------|---|----------------------|--------------------------------------|---------|-------|-----------|-----|-----------|
| This Month | Last Month | Title | Publisher | Remarks | Apple | Atari | Commodore | IBM | Macintosh |
| Entertainment | | | | | | | | | |
| 1. | | <i>Karate Champ</i> | Data East | Martial arts game | • | | • | | |
| 2. | | <i>Kung Fu Master</i> | Data East | Martial arts game | • | | • | | |
| 3. | 1. | <i>Ultima IV</i> | Origin Systems, Inc. | Fantasy game | • | • | • | | |
| 4. | | <i>Silent Service</i> | Microprose | Submarine simulation | • | • | • | • | |
| 5. | | <i>Flight Simulator II</i> | SubLogic | Aircraft simulation | • | • | • | | |
| Education | | | | | | | | | |
| 1. | 2. | <i>Typing Tutor III</i> | Simon & Schuster | Typing instruction program | • | | • | • | • |
| 2. | 1. | <i>Math Blaster!</i> | Davidson | Introductory math program, ages 6-12 | • | • | • | • | |
| 3. | 5. | <i>Homework Helper: Math Word Problems</i> | Spinnaker | Math tutorial, high school level | • | | • | | |
| 4. | | <i>Color Me: The Computer Coloring Kit</i> | Mindscape | Children's artistic tool | • | | • | | |
| 5. | 3. | <i>New Improved MasterType</i> | Scarborough | Typing instruction program | • | • | • | • | |
| Home Management | | | | | | | | | |
| 1. | 2. | <i>The Newsroom</i> | Springboard | Do-it-yourself newspaper | • | | • | • | • |
| 2. | 1. | <i>Print Shop</i> | Broderbund | Do-it-yourself print shop | • | • | • | | |
| 3. | | <i>Print Master</i> | Unison World | Do-it-yourself print shop | • | | • | • | |
| 4. | | <i>Sylvia Porter's Personal Financial Planner</i> | Timeworks | Personal financial package | • | | • | • | |
| 5. | | <i>Bank Street Writer</i> | Broderbund | Word processing program | • | • | • | • | |

Copyright 1986 by Billboard Publications, Inc. Compiled by the Billboard Research Department and reprinted by permission. Data as of 5/10/86 (entertainment) and 5/3/86 (education and home management).

To Our Readers:

COMPUTE! Publications is a part of the ABC Consumer Magazines group of ABC Publishing, Inc. and recently we consolidated many of our operations and moved our Customer Service Department to the New York ABC headquarters. If you have any questions regarding back issues, disk orders, book orders, or how to place an order, call toll free **1-800-346-6767** New York residents should call 212-887-8525.

If you want to order a subscription to COMPUTE!, COMPUTE!'s GAZETTE, COMPUTE!'s GAZETTE DISK, or the COMPUTE! DISK, call **1-800-247-5470** or in Iowa call 1-800-532-1272.

Our Editorial Offices remain in Greensboro, North Carolina. If you wish to submit an article for publication, write us at COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403.

We thank you for your interest and continued support of COMPUTE! Publications.

COMPUTE! Publications, Inc. 

Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies

COMPUTE's Author Guide

Most of the following suggestions serve to improve the speed and accuracy of publication. COMPUTE! is primarily interested in new and timely articles on the Commodore 64/128, Atari, Apple, IBM PC/PCjr, Amiga, and Atari ST. We are much more concerned with the content of an article than with its style, but articles should be clear and well-explained.

The guidelines below will permit your good ideas and programs to be more easily edited and published:

1. The upper left corner of the first page should contain your name, address, telephone number, and the date of submission.
2. The following information should appear in the upper right corner of the first page. If your article is specifically directed to one make of computer, please state the brand name and, if applicable, the BASIC or ROM or DOS version(s) involved. In addition, *please indicate the memory requirements of programs.*
3. The underlined title of the article should start about 2/3 of the way down the first page.
4. Following pages should be typed normally, except that in the upper right corner there should be an abbreviation of the title, your last name, and the page number. For example: Memory Map/Smith/2.
5. All lines within the text of the article must be double- or triple-spaced. A one-inch margin should be left at the right, left, top, and bottom of each page. No words should be divided at the ends of lines. And please do not justify. Leave the lines ragged.
6. Standard typing paper should be used (no erasable, onionskin, or other thin paper) and typing should be on one side of the paper only (upper- and lowercase).
7. Sheets should be attached together with a paper clip. Staples should not be used.
8. If you are submitting more than one article, send each one in a separate mailer with its own tape or disk.
9. Short programs (under 20 lines) can easily be included within the text. Longer programs should be separate listings. *It is essential that we have a copy of the program, recorded twice, on a tape or disk.* If your article was written with a word processor, we also appreciate a copy of the text file on the tape or disk. Please use high-quality 10 or 30 minute tapes with the program recorded on both sides. The tape or disk should be labeled with the author's name, the title of the article, and, if applicable, the BASIC/ROM/DOS version(s). Atari tapes should specify whether they are to be LOADED or ENTERED. We prefer to receive Apple programs on disk rather than tape. Tapes are fairly sturdy, but disks need to be enclosed within plastic or

cardboard mailers (available at photography, stationery, or computer supply stores).

10. A good general rule is to spell out the numbers zero through ten in your article and write higher numbers as numerals (1024). The exceptions to this are: Figure 5, Table 3, TAB(4), etc. Within ordinary text, however, the zero through ten should appear as words, not numbers. Also, symbols and abbreviations should not be used within text: use "and" (not &), "reference" (not ref.), "through" (not thru).

11. For greater clarity, use all capitals when referring to keys (RETURN, TAB, ESC, SHIFT), BASIC words (LIST, RND, GOTO), and three languages (BASIC, APL, PILOT). Headlines and subheads should, however, be initial caps only, and emphasized words are not capitalized. If you wish to emphasize, underline the word and it will be italicized during typesetting.

12. Articles can be of any length—from a single-line routine to a multi-issue series. The average article is about four to eight double-spaced, typed pages.

13. If you want to include photographs, they should be either 5×7 black and white glossies or color slides.

14. We do not consider articles which are submitted simultaneously to other publishers. If you wish to send an article to another magazine for consideration, please do not submit it to us.

15. COMPUTE! pays between \$70 and \$800 for published articles. In general, the rate reflects the length and quality of the article. Payment is made upon acceptance. Following submission (Editorial Department, COMPUTE! Magazine, P.O. Box 5406, Greensboro, NC 27403) it will take from four to eight weeks for us to reply. If your work is accepted, you will be notified by a letter which will include a contract for you to sign and return. *Rejected manuscripts are returned to authors who enclose a self-addressed, stamped envelope.*

16. If your article is accepted and you have since made improvements to the program, please submit an entirely new tape or disk and a new copy of the article reflecting the update. We cannot easily make revisions to programs and articles. It is necessary that you send the revised version as if it were a new submission entirely, but be sure to indicate that your submission is a revised version by writing, "Revision" on the envelope and the article.

17. COMPUTE! does not accept unsolicited product reviews. If you are interested in serving on our panel of reviewers, contact the Review Coordinator for details.

Screen Handler 64

Jeffrey Bailey

This useful utility adds some word processor-like features to the Commodore 64's full-screen editor for use within your own BASIC programs. It also works on the Commodore 128 in 64 mode.

Although it's often taken for granted, the Commodore 64 has one of the best full-screen editors in the business. You can easily move the cursor anywhere on the screen, type characters, and make changes wherever you like. Not all computer owners are so lucky.

As good as it is, however, there are some things that the Commodore 64's excellent screen editor can't do. If you were to draw up a wish list for a new screen editor that would be active when a BASIC program requests keyboard input, it might include these features:

- Switchable insert mode.
- Delete key that draws text *into* the cursor.
- Key to move the cursor to the beginning of input.
- Key to move the cursor to the last character typed.
- Key to erase text from the cursor to the end of the line.

That's quite an impressive list, but there's one more feature that's also desirable: a special key to clear out a single field. In this context, a *field* is simply a screen area of a

certain size in which the user can type characters. For instance, let's say your BASIC program needs to request the date in the format mm/dd/yy. Each two-digit entry could be defined as a field that's two characters in length. Ideally, the user wouldn't need to type the / character between each field. As each two-digit entry was completed, the cursor would move automatically from the end of one field to the beginning of the next.

To take this concept even further, how about setting up multiple fields at different spots on the screen? If each field works as we've described, it wouldn't be necessary to press RETURN after typing data in each field. During the entry process, the user could move freely from field to field until each one is filled in. Then your program could read the entire screenful of data at once.

It may sound like a tall order, but "Screen Handler 64" makes all of this possible.

Entering The Program

Although Screen Handler 64 is written completely in machine language, no knowledge of machine language is needed to use it. To type in Screen Handler 64, simply use the "MLX" machine language entry program listed elsewhere in this issue. Follow the MLX directions carefully. Here are the addresses you need for MLX:

Starting address: C000
Ending address: C397

You can then begin entering the Screen Handler data from Program 1. When you finish entering all the data, be sure to use the MLX Save option to store at least one copy on tape or disk.

To load Screen Handler into memory, use `LOAD "filename",8,1` for disk or `LOAD "filename",1,1` for tape. Although Screen Handler is less than 1K in length, it uses all of the free memory from locations 49152-53247, so you shouldn't do anything to change those locations while the program is active. In addition, you should avoid changing the contents of memory locations 251-255, which Screen Handler uses as well.

Starting Screen Handler

It's quite simple to incorporate Screen Handler into your BASIC programs. After the machine language is in memory, set up a table of the fields you want, then call Screen Handler with a SYS command. Screen Handler takes care of everything else, including moving the input data into string variables. Getting started is actually a three-stage process. Let's look at each one in turn.

The first step after loading Screen Handler is to include the statement `SYS 49152` in your BASIC program. This statement, without any extra parameters,

clears the field table that Screen Handler uses internally.

The next step is to set up each individual field, a job that's also done with a SYS. Here's the general format for the command:

SYS 49155,x,y,length,string\$

Notice that this SYS statement is followed by four parameters. The first two, *x* and *y*, stand for horizontal and vertical screen locations, and can be either numbers or numeric variables. The *x* value must be in the range 0-39, and *y* must be in the range 0-24.

The next parameter, *length*, defines the length of the field in characters. The length parameter can be any numeric value from 1-254. However, keep in mind that the field must not be so long that the screen scrolls when data is typed. If this happens, Screen Handler can't read the input correctly (because the data has moved up one or more lines).

The final parameter, *string\$*, can be any type of string variable, from a simple string such as A\$ to an array element like FI\$(6). Multi-dimensional arrays like FI\$(8,2) can also be used. Screen Handler automatically takes the information from the screen and puts it into this string variable. Note that if the input data is shorter than the field length (if the user types DOG in a ten-character field, for instance), Screen Handler fills the remainder of the string with spaces.

Once you've defined a string within a Screen Handler field, you should not redefine the string until after Screen Handler has been called (see below). This is because Screen Handler sets up pointers to BASIC's string storage space. If you suddenly redefine a string, it may move in memory, confusing the program. After Screen Handler has been called and input has been entered from the keyboard, it is safe to modify the string.

A Better Way To INPUT

Screen Handler permits you to define as many as 50 different fields. When all of the fields are set up, it's time to call Screen Handler. The statement SYS 49158 tells Screen Handler to begin receiving input from the fields you have defined.

At this point, the up and down cursor keys move the cursor from field to field, *not* from line to line as usual. The RETURN key has the same effect as cursor down, moving you forward (down) to the next field. The left and right cursor keys work normally, but only within a field. That is, these keys move the cursor left and right inside the field as usual. But to move to another field, you must press cursor up/down or RETURN.

Pressing CLR/HOME moves the cursor to the beginning of the current field. SHIFT-CLR/HOME erases the field and homes the cursor. The INST/DEL key works normally, but acts only on the current field. The CTRL-I key combination (hold down CTRL and press I) switches Screen Handler in and out of insert mode. The border color changes to indicate when insert mode is active.

Pressing CTRL-D deletes text by pulling it into the present cursor position. CTRL-E erases every character from the cursor to the end of the field. To move the cursor to the last character typed in a field, press CTRL-N.

That takes care of the entire wish list except for the most important part—storing all of the input data in variables. When all of the data is entered in all of the fields, press SHIFT-RETURN. Screen Handler enters the entire screen at once.

Practical Demonstration

Program 2, "Screen Handler Demo," is a simple BASIC program that illustrates how to use Screen Handler. Here's an explanation of how it works.

Line 10 loads Screen Handler from disk, and lines 20-60 create a screen display that outlines the fields visually. The DATA statement in line 70 contains *x*, *y*, and *length* values for defining the fields. Line 90 clears the field table and prepares Screen Handler for use. Lines 100-140 set up a simple loop to read in the values and set them up in Screen Handler's table.

Note that although an array is used in line 130, the program contains no DIM statement. If BASIC encounters an array that was not previously dimensioned, it auto-

matically dimensions the array for 11 elements (numbered 0-10). Since Screen Handler uses some of BASIC's built-in routines, this feature is available as usual. However, it would be better programming practice to explicitly dimension the array variables—and the DIM is required if you want to use more than 11 array elements. In this case, the DIM statement must precede the SYS 49155 statement that assigns the array element to a field. In Program 2, the DIM statement could be placed anywhere before line 100.

Line 160 tells Screen Handler to start accepting data. Again, notice that the string variables must not be modified between the time that the fields are defined (line 130) and Screen Handler is called (line 160). Once the data has all been entered and you press SHIFT-RETURN to accept the fields, lines 170-210 clear the screen and print out the information. At this point in the program, it becomes safe to modify the string variables if needed.

With a little bit of practice, you can write very professional programs in BASIC with Screen Handler's help. Experiment with programs of your own.

Program 1: Screen Handler 64

Please refer to the "MLX" article in this issue before entering the following listing.

```
C000:4C 09 C0 4C 18 C0 4C 73 96
C008:C0 A9 FB 85 FB A9 00 85 38
C010:FD A9 C4 85 FE 85 FC 60 54
C018:A5 FB C9 FE F0 54 18 69 07
C020:05 85 FB 20 FD AE 20 9E 90
C028:B7 8A A0 00 91 FB 20 FD F6
C030:AE 20 9E B7 8A A0 01 91 CA
C038:FB 20 FD AE 20 9E B7 8A DF
C040:C9 FF D0 03 38 E9 01 48 A4
C048:A0 02 91 FB 20 FD AE 20 03
C050:8B B0 85 9E 84 9F 20 A3 E4
C058:B6 68 20 75 B4 A0 02 B9 90
C060:61 00 91 9E 88 10 F8 A0 C5
C068:03 A5 62 91 FB C8 A5 63 EB
C070:91 FB 60 A5 FB C9 FB D0 EF
C078:01 60 20 92 C2 A9 80 8D 0B
C080:8A 02 A0 00 8C 0E 03 84 03
C088:B6 B1 FD 99 A7 00 C8 00 BA
C090:05 D0 F6 A9 01 85 02 A4 09
C098:A7 A6 A8 18 20 F0 FF 20 13
C0A0:35 C2 8C A7 02 8E A8 02 17
C0A8:20 00 C1 A0 00 AD 86 02 42
C0B0:91 BD B1 B4 49 80 91 B4 10
C0B8:20 EA FF C9 00 F0 F9 A0 78
C0C0:00 AA B1 B4 49 80 91 B4 92
C0C8:20 41 C1 A2 01 E4 B6 D0 E6
C0D0:0D C9 1D F0 09 C9 00 F0 5E
C0D8:05 48 20 77 C2 68 20 D2 35
C0E0:FF A9 00 85 D4 20 35 C2 79
C0E8:CC A7 02 D0 08 EC A8 02 4F
C0F0:D0 03 4C 9F C0 E6 02 A5 6A
C0F8:A9 C5 02 B0 A2 4C F9 C1 08
```


DISK SALE!

PREMIUM QUALITY! LIFETIME WARRANTY!

Made by top makers, not "low-end" or "seconds".
We buy millions from the factories! We can't
print the maker, but you'll recognize them as
the highest quality premium disks made!
Guaranteed 100% error free. SATISFACTION
GUARANTEED OR YOUR MONEY BACK! BULK
PRICES

| Prices are per disk | 100 | 500 | 1K | 5K | 10K | 25K | 100K | 1K | 5K |
|---------------------------------------|------|------|------|------|------|------|------|------|------|
| For the Apple II, II+, IIx, IIC, IICx | 45¢ | 42¢ | 39¢ | 36¢ | 33¢ | 30¢ | 27¢ | 24¢ | 21¢ |
| SS/DD for single side computers | 50¢ | 47¢ | 44¢ | 41¢ | 38¢ | 35¢ | 32¢ | 29¢ | 26¢ |
| SS/DD for IBM PC and compatibles | 55¢ | 52¢ | 49¢ | 46¢ | 43¢ | 40¢ | 37¢ | 34¢ | 31¢ |
| SS/DD for IBM AT | 1.00 | 94¢ | 89¢ | 84¢ | 79¢ | 74¢ | 69¢ | 64¢ | 59¢ |
| SS/DD for IBM AT w/AT | 1.35 | 127¢ | 122¢ | 117¢ | 112¢ | 107¢ | 102¢ | 97¢ | 92¢ |
| 3 1/2" single side for Macintosh | 1.35 | 127¢ | 122¢ | 117¢ | 112¢ | 107¢ | 102¢ | 97¢ | 92¢ |
| 3 1/2" double side for Macintosh | 2.50 | 235¢ | 230¢ | 225¢ | 220¢ | 215¢ | 210¢ | 205¢ | 200¢ |
| 5 1/4" SLEEVES, 244 white w/SLV | 5¢ | 45¢ | 44¢ | 43¢ | 42¢ | 41¢ | 40¢ | 39¢ | 38¢ |
| 5 1/4" SLEEVES, white Tyvek HTV | 7¢ | 67¢ | 66¢ | 65¢ | 64¢ | 63¢ | 62¢ | 61¢ | 60¢ |

Order BULK (N-pak) with no sleeves or box; BOXED IN TENS (D-pak) or
JUMBO BOXED IN 25's (J-pak) with sleeves, labels, re-usable box and
write protect tabs. Add 2¢ per disk for D-pak, 10¢ for J-pak. Micro-disks are
poly-bagged.

INCREDIBLE PRICES ON ACCESSORIES!

UNIFILE-10 library cases, clear plastic, holds 10 disks #F10 99¢
UNIFILE-100 holds 100 disks, with lock & key, removable top #F100 513.88
UNIFILE-70 holds 70 disks, flip-top #F70 58.88
UNIPAK MAILERS, cardboard, holds 25 disks #UPAK 10 for 55¢
RIGID PAK MAILERS, hard plastic, for mailing or purse, holds 5 disks #RPAK 99¢

HOW TO ORDER: Order by phone or mail, pay by MC-Visa-Amex-COD,
or send check with order, minimum order \$20. \$3 for shipping, \$2 if COD.
Canada/W.A. shipping \$5. Open accounts avail for schools with good
credit, minimum purchase order \$100. FOB Unitech. All orders must include
daytime phone and STREET address. Send for FREE CATALOG.

UNITECH

IN MASS (617) 317-1100
200 HURLEY STREET
CAMBRIDGE, MA 02141

(800)343-0472



FREE COLOR MONITOR



With purchase of Computer

SPECIALIST IN

HOME AND BUSINESS COMPUTERS

612-938-3161

821 Main Street, Hopkins, MN

STATE-OF-THE-ART MAGNETIC MEDIA

5 1/4" DISKETTES

- With Hub Rings
- Write Protect Tabs
- Envelopes
- User ID Labels
- In Factory Sealed Poly Packs of 10

(YOU GET EVERYTHING BUT THE BOX)

Prices are per Disk

| QTY. | 50 | 100 | 500 | 1000 |
|------|-----|-----|-----|------|
| SSDD | .59 | .53 | .50 | .47 |
| LSDD | .62 | .58 | .55 | .52 |

Library Case Holds 15 Diskettes..... Only \$1.00!
plus 50¢ S&H

The 100 File Only \$10.95 plus \$2.00 S&H
100% ERROR FREE — LIFETIME WARRANTY
Min. order \$25.00. Add 10% for less than 50
diskettes. Shipping and Handling: \$4.00 per 100
diskettes. Reduced shipping for larger quantities.
C.O.D. add \$4.00. Cash or certified check.
Continental USA

ccp

Precision Data Products
P.O. Box 8367, Grand Rapids, MI 49518
(616) 452-3457 • Michigan 1-800-632-2468
Outside Michigan 1-800-258-0028

C100:98 48 8A 48 A9 00 85 B4 C4
C108:85 B5 68 85 FF A2 00 E4 90
C110:FF F0 0F A5 B4 18 69 28 0D
C118:90 02 E6 B5 85 B4 E8 4C B9
C120:0F C1 68 85 FF A5 B4 18 19
C128:65 F1 90 02 E6 B5 85 B4 5E
C130:85 BD A5 B5 18 69 D8 85 93
C138:BE A5 B5 18 69 04 85 B5 D8
C140:60 8A C9 93 D0 05 68 68 DC
C148:4C DE C2 C9 13 D0 05 68 EC
C150:66 4C 93 C0 C9 91 D0 05 D4
C158:68 68 4C D5 C1 C9 11 D0 39
C160:05 68 68 4C F9 C1 C9 9D 5A
C168:D0 06 68 68 4C 22 C2 60 3A
C170:C9 0D D0 05 68 68 4C F9 FD
C178:C1 C9 94 D0 08 20 77 C2 61
C180:68 68 4C 9F C0 C9 14 D0 FB
C188:05 68 68 4C C8 C2 C9 09 68
C190:D0 18 A5 B6 D0 0A E6 B6 D5
C198:EE 20 D0 68 68 4C 9F C0 B0
C1A0:C6 B6 CE 20 D0 68 68 4C 56
C1A8:9F C0 C9 04 D0 06 20 A7 C2
C1B0:C2 A9 00 60 C9 8D D0 05 31
C1B8:68 68 4C F3 C2 C9 05 D0 6B
C1C0:08 20 41 C3 68 68 4C 9F D1
C1C8:C0 C9 0E D0 E6 20 59 C1 C0
C1D0:68 68 4C 9F C0 A5 FD C9 88
C1D8:08 D0 05 A5 FB 18 69 05 A3
C1E0:38 E9 05 85 FD A0 00 A9 10
C1E8:A7 85 71 84 72 B1 FD 91 FF
C1F0:71 C8 C0 05 D0 F7 4C 93 5A
C1F8:C0 A5 FD 05 FB D0 02 A9 33
C200:FB 18 69 05 85 FD A0 00 6C
C208:A9 A7 85 71 84 72 B1 FD 63
C210:91 71 C8 C0 05 D0 F7 AD E8
C218:0E 03 C9 AA D0 01 60 4C E0
C220:93 C0 A9 01 C5 02 90 03 3F
C228:4C 9F C0 C6 02 A9 9D 20 52
C230:D2 FF 4C 9F C0 38 20 F0 BA
C238:FF 98 C9 28 90 04 38 E9 8E
C240:28 A8 60 20 35 C2 20 00 07
C248:C1 A5 B4 85 9E A5 B5 85 83
C250:9F A5 BD 85 AE A5 BE 85 2E
C258:AF A4 A7 A6 A8 20 00 C1 C5
C260:A5 9E 38 E5 B4 85 FF A5 27
C268:A9 38 E5 FF C9 02 B0 03 48
C270:68 68 60 38 E9 02 60 20 0C
C278:43 C2 A8 B1 9E C8 91 9E 5A
C280:88 B1 AE C8 91 AE 88 88 F9
C288:C0 FF D0 EF C8 A9 20 91 46
C290:9E 60 A0 00 A9 DA 85 B4 0A
C298:84 B5 B1 B4 29 7F 91 B4 6E
C2A0:C8 C8 C0 18 D0 F4 60 20 91
C2A8:43 C2 18 69 02 85 FF A0 E0
C2B0:01 B1 9E 88 91 9E C8 B1 C9
C2B8:AE 88 91 AE C8 C8 C4 FF C7
C2C0:D0 EF 88 A9 20 91 9E 60 3B
C2C8:A5 02 C9 01 D0 03 4C 9F B5
C2D0:C0 C6 02 A9 9D 20 D2 FF 56
C2D8:20 A7 C2 4C 9F C0 A6 A8 6B
C2E0:A4 A7 20 00 C1 A0 00 A9 E0
C2E8:20 91 B4 C8 C4 A9 D0 F9 6E
C2F0:4C 93 C0 A5 FB 85 FD A9 8F
C2F8:AA 8D 0E 03 20 F9 C1 A4 3A
C300:A7 A6 A8 20 00 C1 A0 00 64
C308:B1 B4 20 26 C3 91 AA C8 7E
C310:C4 A9 D0 F4 20 F9 C1 A5 DF
C318:FD F0 03 4C FF C2 A5 B6 0D
C320:F0 03 CE 20 D0 60 29 7F 96
C328:C9 20 B0 04 18 69 40 60 3A
C330:C9 40 B0 01 60 C9 60 B0 6E
C338:04 18 69 20 60 18 69 40 6D
C340:60 20 43 C2 C9 00 F0 10 D4
C348:18 69 02 85 FF A0 00 A9 FA
C350:20 91 9E C8 C4 FF D0 F9 6E
C358:60 A6 A8 A4 A7 20 00 C1 98
C360:A4 A9 88 B1 B4 C0 00 F0 6A
C368:29 C9 20 F0 F5 C8 C4 A9 10
C370:D0 01 88 98 48 A4 A7 A6 06
C378:A8 18 20 F0 FF A9 01 85 9B
C380:02 68 85 FF A0 00 A9 1D 49
C388:20 D2 FF C8 E6 02 C4 FF 2A
C390:D0 F4 60 00 00 00 00 00 C9

Program 2: Screen Handler Demo

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing in Programs" in this issue of COMPUTE!.

```

PB 10 IF LD=0 THEN LD=1:LOAD "
      SCREEN HANDLER",8,1
SP 20 PRINT "{CLR}":PRINT "****
      SCREEN HANDLER ***":PRI
      NT
BQ 30 PRINT "NAME:{5 SPACES}[
      {20 SPACES}]:PRINT
HC 40 PRINT "ADDRESS:
      {2 SPACES}[[20 SPACES]]"
      :PRINT
PG 50 PRINT "CITY/ST:
      {2 SPACES}[[15 SPACES]]
      {2 SPACES}[[2 SPACES]]":
      PRINT
MS 60 PRINT "PHONE:{4 SPACES}1
      -({3 SPACES})-{3 SPACES}
      -{4 SPACES}"
EM 70 DATA 11,3,20,11,5,20,11,
      7,15,30,7,2,13,9,3,18,9,
      3,22,9,4
RK 80 REM *** CLEAR TABLE ***
CA 90 SYS 49152
KM 100 FOR A=1 TO 7
MR 110 READ X,Y,L
DQ 120 REM *** SET UP TABLE **
      *
JB 130 SYS 49155,X,Y,L,A$(A)
MM 140 NEXT
QS 150 REM *** CALL SCREEN HAN
      DLER ***
XD 160 SYS 49158
CJ 170 PRINT "{CLR}"
EB 180 PRINT A$(1)
QA 190 PRINT A$(2)
SQ 200 PRINT A$(3);",",A$(4)
SJ 210 PRINT "1-({A$(5);")-";
      A$(6);"-";A$(7)
  
```

COMPUTE!

TOLL FREE

Subscription

Order Line

1-800-247-5470

In IA 1-800-532-1272

Attention Programmers

COMPUTE! magazine is currently looking for quality articles on Commodore, Atari, Apple, and IBM computers (including the Commodore Amiga and Atari ST). If you have an interesting home application, educational program, programming utility, or game, submit it to COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Or write for a copy of our "Writer's Guidelines."

Atari Sound Development System

Michael Ryder

This versatile program lets you design sounds onscreen with a joystick and the keyboard, taking advantage of virtually every feature built into the Atari's sound chip—including some features which are rarely exploited. The custom sounds can then be saved on disk, played back, or added to your own BASIC programs. For all Atari 400/800, XL, and XE computers with at least 40K RAM and a disk drive.

Ever since the Atari was first introduced in 1979, its sound capabilities have always played second fiddle to its graphics. In fact, even many Atari owners are unaware that the sound chip inside their computer has switchable high-pass filters, optional 16-bit frequency resolution, and an adjustable clock rate for modifying the frequency range. Part of the problem is that Atari BASIC's SOUND statement doesn't begin to touch these capabilities; they're accessible only with PEEK and POKE or machine language.

The three programs included here, collectively known as the "Atari Sound Development System," make it easier for you to take advantage of these features—or get acquainted with them in the first place. The main program, "Sound Editor," is a utility that puts the full range of Atari sound capabilities at your command with keyboard and joystick controls. It also lets you design sounds with ADSR envelopes—a feature that, as we'll see in a moment, isn't even built into the Atari sound chip.

Two additional programs,

"Sound Player" and "Sound Program Writer," let you play back the sounds you create with the Sound Editor or automatically generate stand-alone programs that can be converted into subroutines of your own BASIC programs.

Developing Sounds

To get started, type in and save Program 1 below. The Sound Editor is the main program that lets you develop, modify, save, and load sounds.

When you type RUN, at first you'll see nothing but a black screen and hear a few beeps. The beeps signal that everything is running normally while the Sound Editor sets itself up. After a short delay, you'll see the Main Menu, which leads to several submenus for various functions:

MAIN MENU

- 1—Develop sounds
- 2—Save/Load/Del/Dir sound envelopes
- 3—EXIT PROGRAM

Your choice (1-3): ?

Option 1 is the gateway into the main part of the program. Option 2 leads to the Input/Output Menu. Option 3 stops the program and exits to BASIC. Let's tackle option 1 first, since that's the meat of the Sound Editor.

When you press 1, the Sound Menu pops up:

SOUND MENU

- 1—Envelope Editor, Voice 0
- 2—Envelope Editor, Voice 1
- 3—Envelope Editor, Voice 2
- 4—Envelope Editor, Voice 3
- 5—Play Voices Menu
- 6—Clear all voices
- 7—MAIN MENU

Your choice (1-7): ?

Options 6 and 7 are fairly obvious: 6 resets all four voices, discarding any existing values that may have been entered, and 7 returns to the Main Menu shown above. The other options let you design, modify, and play sounds in numerous ways.

Options 1-4 let you use a joystick to design individual sound envelopes for any of the four voices. Each Envelope Editor screen shows a graphic display of the current envelope and also indicates the pitch value assigned to that voice (see photos). Since envelopes aren't actually built into the Atari sound chip, but instead are handled by this program, let's backtrack for a moment and explain how they work.

Attack And Retreat

One of the many characteristics which distinguish different sounds is the shape of their ADSR envelopes. ADSR stands for *attack*, *decay*, *sustain*, and *release*. These are the four stages of volume changes that occur during a sound's duration.

Attack is the initial rise in volume to the sound's peak volume. Decay is the decrease in volume that follows the peak. Sustain is the period in which the sound continues to be audible. And release is the final drop in volume to silence. Photo 1 is a typical ADSR envelope.

By changing the shape of this envelope, you can vary the effect of the sound. For instance, a percussive sound has an almost instantaneous attack, very short decay and

sustain, and a fairly sharp release (Photo 2).

If you pluck a guitar string and let it resonate, the attack is some-

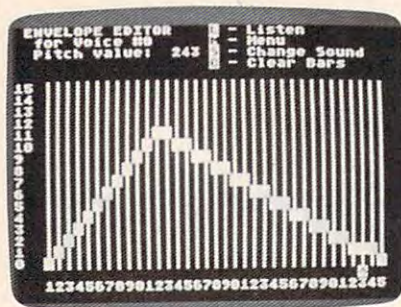


Photo 1: An attack-decay-sustain-release envelope created with the Sound Editor.



Photo 2: A sharp attack and fast release is typical of percussion instruments.

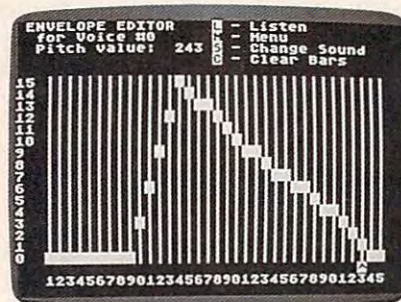


Photo 3: Most musical instruments have a more gentle attack and release.

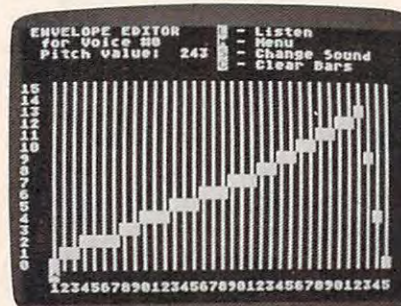


Photo 4: This envelope makes a backward sound.

what less steep and the sustain/release is much more gradual (Photo 3).

Since most real-world sounds have similar envelopes, sounds with a gradual attack and a rapid sustain/release often seem backward and artificial (Photo 4).

You can design almost any kind of ADSR envelope with the Sound Editor. At the bottom of the Envelope Editor screen for each voice is a joystick-controlled cursor. By pushing the joystick left or right, you can move the cursor horizontally to pick a position within the envelope. To set a volume level for that position, press the joystick button, then move the cursor up and down with the stick. When the cursor is at the desired level, press the joystick button again. The level is marked with a white block, and you can move the cursor left or right again to pick the next position.

If you change your mind and want to reset a volume level within the envelope, just move back to that position with the joystick and press the button as before. You can clear out the entire envelope by pressing C (Clear bars).

Other keyboard commands available on the Envelope Editor screens are L (Listen to the envelope), M (return to the Sound Menu), and S (change Sound). If you finish one voice's envelope and want to design an envelope for a different voice, press M for the Sound Menu, then select 1-4 to switch to the other voice's Envelope Editor.

Controlling Sounds

When you press S (change Sound) on an Envelope Editor screen, you get another screen that offers a wide range of control over the POKEY chip, which is responsible for Atari sound. Most of these controls are inaccessible from BASIC without PEEK and POKE. The control screen displays this information:

Modify/Mix Sounds

PRESS TRIGGER TO LISTEN TO ALL VOICES

Switch clock from 64KHz to 15KHz :0
Hi-pass filter on ch.1, clock by 3 :0
Hi-pass filter on ch.0, clock by 1 :0
Join channels 3 and 2 :0
Join channels 1 and 0 :0
Clock channel 2 with 1.79 MHz :0

Clock channel 0 with 1.79 MHz :0
Change from 17 to 9 bit poly :0
CHANNEL :0
VOLUME :8
DISTORTION :10
FREQUENCY :243
STATUS (0/1) :ON
—Press X to Exit Back to Editor—

Press the joystick trigger to play the voices; press it again to stop. The X key returns you to the Envelope Editor.

The other features on the control screen let you change various POKEY settings for the voices. At the left and right of this screen, you'll notice a pair of pointers (greater-than and less-than signs). You can move the pointers up and down the screen with the A and Z keys, respectively. This is how you select a certain control you want to change.

The first eight controls (Switch clock from 64KHz to 15KHz through Change from 17 to 9 bit poly) can be set to either 0 or 1. When the control is set to 0, it is off; when it's set to 1, it's on. For example, to switch on the control for Hi-pass filter on ch.0, clock by 1, you'd move the pointers to that line with the A or Z key, then type 1. To turn it off, you'd type 0. You can turn any of the controls on or off in any combination.

The last five controls on this screen (CHANNEL through STATUS) work a little differently. After selecting one with the pointers, press the space bar. A question mark prompts you to enter a new value. Type in the new value and press RETURN. The allowable ranges are CHANNEL (0-3), VOLUME (0-15), DISTORTION (0-14), FREQUENCY (0-255), and STATUS (0=ON or 1=OFF). These correspond to the parameters in the BASIC SOUND statement, except for STATUS, which turns the current voice on or off so you can mix the different voices.

The first eight controls, however, aren't accessible with the SOUND statement; they POKE values into certain memory locations which directly control the POKEY chip. Perhaps the best way to learn what these controls do is to run the Sound Editor and simply experiment. For a more technical explanation, read the following section.

Playing POKEY

The POKEY chip is accessed through nine memory locations known as AUDF1 (53760), AUDF2 (53762), AUDF3 (53764), AUDF4 (53766), AUDC1 (53761), AUDC2 (53763), AUDC3 (53765), AUDC4 (53767), and AUDCTL (53768). AUDCTL (AUDio ConTrol) controls both the four AUDio Frequency registers (which set the frequencies of the voices) and the four AUDio Control registers (which set the volumes and distortions of the voices). Each of the eight bits in AUDCTL (0-7) controls some aspect of the sound produced. These eight bits correspond to the eight controls in the Sound Editor:

| Bit | Decimal | Description |
|-----|---------|---|
| 0 | 1 | Switch main clock base from 64 KHz to 15 KHz. |
| 1 | 2 | Insert high-pass filter into channel 2, clocked by channel 4. |
| 2 | 4 | Insert high-pass filter into channel 1, clocked by channel 2. |
| 3 | 8 | Join channels 4 and 3 (16 bit). |
| 4 | 16 | Join channels 2 and 1 (16 bit). |
| 5 | 32 | Clock channel 3 with 1.79 MHz. |
| 6 | 64 | Clock channel 1 with 1.79 MHz. |
| 7 | 128 | Make the 17-bit poly counter into 9-bit poly. |

The clock bits (0, 5, and 6) speed up or slow down clock timers, making higher or lower frequency ranges possible. Setting the voices to 1.79 MHz with bits 5 and 6 will produce a much higher sound. The 64 KHz clock will produce lower sounds, and the 15 KHz clock the lowest.

The filter bits (1 and 2), when turned on, allow only frequencies higher than the clock value to pass through.

Bits 3 and 4 each join two of the channels together to permit 16-bit frequency resolution, allowing greater range (nine octaves instead of five) and pitch accuracy. Normally, the POKEY chip uses only 8-bit frequency resolution.

Finally, bit 7 makes the 17-bit poly counter into a 9-bit poly counter. If this bit is set when using distortion, the distortion pattern becomes more obvious.

Playing The Sounds

Let's go back to the Sound Menu, where there's one more option we haven't covered:

SOUND MENU

- 1—Envelope Editor, Voice 0
- 2—Envelope Editor, Voice 1
- 3—Envelope Editor, Voice 2
- 4—Envelope Editor, Voice 3

5—Play Voices Menu

6—Clear all voices

7—MAIN MENU

Your choice (1-7): ?

After editing all four voices, you might want to hear what they sound like alone or in unison. Option 5 brings up the Play Voices Menu, which offers all 15 combinations of the four voices at a single keystroke:

PLAY VOICES MENU

— Voice Numbers —

- A—0,1,2,3 B—0,1,2
- C—0,1,3 D—0,1
- E—0,2,3 F—0,2
- G—0,3 H—1,2,3
- I—1,2 J—1,3
- K—2,3 L—0
- M—1 N—2
- O—3 P—PAST MENU

You'll notice, however, that the sound plays very slowly when you use this option. This is because of the memory consumed by the program, BASIC, and the routine that checks which voices are played in each combination. But don't fret—the sounds are much faster (and better) when played with a shorter BASIC program, such as Program 2, Sound Player. This program simply loads and plays any sound created with the Sound Editor.

Program 3, Sound Program Writer, takes a sound created with the Sound Editor, compacts it, and automatically writes a BASIC program to play it. During the compacting phase, all unused bars at the ends of envelopes are chopped off. In addition, a menu asks you to choose which voices should be included, and all unused voices are omitted from the resulting program. So, for instance, if you design a sound that uses only one of the four voices, you can tell Sound Program Writer to discard the unused voices and use only the one you edited. This makes the sound play noticeably faster.

To add the sound to a BASIC program of your own, simply use the program created by Sound Pro-

gram Writer as a subroutine. It's saved on disk in ASCII format and can be merged into any program with ENTER.

Disk Operations

Before you can play a sound with the Sound Player or convert it into a program with the Sound Program Writer, you have to save it on disk with the Sound Editor. To do this, select option 2 on the Main Menu (Save/Load/Del/Dir sound envelopes). This brings up the Input/Output Menu:

INPUT/OUTPUT MENU

- 1—Directory of *.SND files
 - 2—Save envelopes file
 - 3—Load envelopes file
 - 4—Delete envelope file
 - 5—MAIN MENU
- Choice: ?

Pick option 2 to save a finished sound on disk. When you type in the filename, don't use an extender; the Sound Editor automatically appends the .SND extender to all envelope files. The same is true when loading a file with option 3 or deleting a file with option 4.

Option 1 calls a directory of all files on the disk with the .SND extender, and option 5 returns to the Main Menu. You can also abort any save, load, or delete operation by typing X and pressing RETURN at the filename prompt.

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!.

Program 1: Sound Editor

```

NC 10 GRAPHICS 0:POKE 82,0:P
   OKE 710,0:POKE 752,1
BA 20 GOTO 3020
LN 30 REM PRIMARY MENU PAGE
IJ 40 COLOR 1:CLOSE #1:OPEN
   #1,4,0,"K:"
OP 50 POKE 89,SCREEN1:POKE 1
   06,SCREEN1+4:POKE DL+5
   ,SCREEN1:GRAPHICS 0:PO
   KE 710,0:POKE 712,0:PO
   KE 752,1
MH 60 POSITION 0,0:?"{Q}
   {38 R}{E}":FOR D=1 TO
   11:POSITION 0,D:?"{I}":
   POSITION 39,D:?"{I}":NE
   XT D
CB 70 POSITION 0,12:?"{Z}
   {38 R}{C}":
EL 80 POSITION 9,2:?"
   {3 SPACES}MAIN ME
   NU{3 SPACES}":
PF 90 POSITION 1,4:?"1 - De
   velop sounds"
HO 100 POSITION 1,5:?"2 - S
   ave/Load/Del/Dir soun
   d envelopes"
AK 110 POSITION 1,7:?"3 - E
   XIT PROGRAM"
  
```



```

NI 120 POSITION 9,9:?"Your
choice (1-3) : ?"
NE 130 GET #1,K:POSITION 29,
9:?" CHR$(K)
BH 140 IF K<49 OR K>51 THEN
POSITION 9,9:?"YOUR
CHOICE (1-3) : "FOR
D=1 TO 10:NEXT D:GOT
O 120
JN 150 ON K-48 GOTO 180,2110
,170
GD 160 GOTO 120
FL 170 GRAPHICS 0:POKE 82,2:
END
BL 180 POKE 710,0:COLOR 1:CL
OSE #1:OPEN #1,4,0,"K
:"
KH 190 POKE 89,SCREEN1:POKE
106,SCREEN1+4:POKE DL
+5,SCREEN1:?" CHR$(125
)
HP 200 POKE 710,0:POKE 712,1
28
DJ 210 POSITION 0,0:?"(Q)
{38 R}{E}":FOR D=1 TO
13:POSITION 0,D:?" :
":POSITION 39,D:?" :
NEXT D:POSITION 0,14:
?" {Z}{38 R}{C}"
KH 230 POSITION 13,2:?" SOL
ID MENU"
LC 240 POSITION 5,4:?"1 - E
nvelope Editor, Voice
0"
LG 250 POSITION 5,5:?"2 - E
nvelope Editor, Voice
1"
LK 260 POSITION 5,6:?"3 - E
nvelope Editor, Voice
2"
LO 270 POSITION 5,7:?"4 - E
nvelope Editor, Voice
3"
FL 280 POSITION 5,8:?"5 - P
lay Voices Menu"
HD 290 POSITION 5,9:?"6 - C
lear all voices"
EF 300 POSITION 5,10:?"7 -
MAIN MENU"
AE 310 POSITION 6,12:?"Your
choice (1-7) : ?"
PM 320 GET #1,K:POSITION 26,
12:?" CHR$(K)
AL 330 IF K-48<1 OR K-48>7 T
HEN POSITION 6,12:?"
YOUR CHOICE (1-7) :
":FOR D=1 TO 10:NEXT
D:GOTO 310
EE 340 CH=K-49
HE 350 ON K-48 GOTO 400,400,
400,400,1820,380,40
GOTO 310
BB 360
IH 370 REM CLEAR ALL VOICES
OC 380 FOR X=0 TO 3:FOR Y=1
TO 35:S(X,Y)=0:NEXT Y
:NEXT X:GOTO 310
NL 390 REM MAIN ROUTINE
JB 400 POKE 89,SCREEN2:POKE
106,SCREEN2+4:POKE DL
+5,SCREEN2:COLOR 160
HP 410 POSITION 13,1:?" CH
BI 420 POSITION 16,2:?" SD(CH
,0)
JN 430 FOR A=1 TO 35:R=S(CH,
A)
HM 440 Y=20-R:X=A+2
LF 450 PLOT X,Y
BK 460 NEXT A
DD 480 OLD=3:POS=OLD
KN 490 COLOR 222:PLOT POS,21
LP 500 REM KEYBOARD CHECKER
CO 510 POKE 764,255
BI 520 S=STICK(0):K=0

AG 530 IF PEEK(764)=255 AND
S=15 AND S<>11 AND S<
>7 AND STRIG(0)<>0 TH
EN 520
NM 540 IF PEEK(764)=255 THEN
600
EF 550 GET #1,K
BC 560 IF CHR$(K)="L" THEN 8
70
JC 570 IF CHR$(K)="M" THEN 1
750
FF 580 IF CHR$(K)="C" THEN 9
00
LK 590 IF CHR$(K)="S" THEN C
OLOR 32:PLOT POS,21:G
OTO 930
GE 600 IF STRIG(0)=0 THEN 72
0
NE 610 IF S=15 OR S<>11 AND
S<>7 AND STRIG(0)=1 T
HEN 520
NB 620 IF NOT ((S=11 AND PO
S=3) OR (S=7 AND POS=
37)) THEN 660
GI 630 IF (S=11 AND POS=3) T
HEN POS=37
DO 640 IF (S=7 AND POS=37) T
HEN POS=3
HC 650 GOTO 680
MH 660 POS=POS+1*(POS<38 AND
S=7)
LP 670 POS=POS-1*(POS>3 AND
S=11)
HM 680 COLOR 222:POKE 53279,
0:PLOT POS,21
BK 690 COLOR 32:PLOT OLD,21
KF 700 OLD=POS
GI 710 GOTO 520
EE 720 REM FILL IN THE BARS
EA 730 FOR S=15 TO 0 STEP -1
:SOUND 0,50,10,S:NEXT
S
MC 740 P=POS-2:V=S(CH,P):X=P
OS:Y=20-V:Y2=Y
EA 750 S=STICK(0):IF S<>14 A
ND S<>13 AND STRIG(0)
=1 THEN 750
GO 760 IF STRIG(0)=0 THEN 84
0
BO 770 V=V+1*(S=14 AND V<15)
OH 780 V=V-1*(S=13 AND V>0)
BL 790 Y=20-V
DB 800 COLOR 124:PLOT X,Y2
GA 810 COLOR 160:PLOT X,Y:Y2
=Y
CP 820 POKE 53279,0
HA 830 GOTO 750
NK 840 S(CH,P)=V
OH 850 FOR S=0 TO 15:SOUND 0
,255,10,S:NEXT S:SOUN
D 0,0,0,0
GO 860 GOTO 520
FC 870 REM LISTEN
JP 880 POKE 53768,BYTE:POKE
53760,SD(CH,0):FOR I=
1 TO 35:R=S(CH,I):POK
E 53761,SD(CH,I)+R:NE
XT I
HE 890 POKE 53761,160:POKE 7
64,255:GOTO 510
ON 900 REM CLEAR THE BARS
CB 910 COLOR 124:FOR A=35 TO
1 STEP -1:R=S(CH,A):
X=A+2:Y=(20-R):PLOT X
,Y:NEXT A:COLOR 32:PL
OT POS,21
FI 920 FOR I=1 TO 35:S(CH,I)
=0:NEXT I:GOTO 400
DF 930 REM CHANGE DISTORTION
BB 940 POKE 82,2:POKE 764,25
5:POKE 89,SCREEN1:POK
E 106,SCREEN1+4:POKE
DL+5,SCREEN1:POKE 752
,1:?" CHR$(125)
FC 950 POKE 53768,0:CHAN=CH
AC 960 POSITION 0,0:?"Modif
y/Mix Sounds":POKE 75
2,1
NH 970 POSITION 0,1:?" PRES
S TRIGGER TO LISTEN T
O ALL VOICES"
DB 980 POSITION 1,2:?"Switc
h clock from 64KHz to
15KHz {3 SPACES}:";BI
T(0)
KC 990 POSITION 1,3:?"Hi-pa
ss filter on ch.1, cl
ock by 3 :";BIT(1)
MA 1000 POSITION 1,4:?"Hi-p
ass filter on ch.0,
clock by 1 :";BIT(2)
NC 1010 POSITION 1,5:?"Join
channels 3 and 2
{14 SPACES}:";BIT(3)
NB 1020 POSITION 1,6:?"Join
channels 1 and 0
{14 SPACES}:";BIT(4)
PF 1030 POSITION 1,7:?"Cloc
k channel 2 with 1.7
9 MHz {6 SPACES}:";BI
T(5)
PB 1040 POSITION 1,8:?"Cloc
k channel 0 with 1.7
9 MHz {6 SPACES}:";BI
T(6)
OL 1050 POSITION 1,9:?"Chan
ge from 17 to 9 bit
poly {7 SPACES}:";BIT
(7)
DP 1060 POSITION 1,10:?"CHA
NNEL {4 SPACES}:";CH
AN
BB 1070 POSITION 1,11:?"VOL
UME {5 SPACES}:";VO
(CHAN)
DE 1080 POSITION 1,12:?"DIS
TORTION : ";SD(CHAN,
1)/16
EC 1090 POSITION 1,13:?"FRE
QUENCY : ";SD(CHAN,
0)
FE 1100 POSITION 1,14:?"STA
TUS {0/1} : ";IF STAT
(CHAN)=1 THEN ? "ON
":GOTO 1120
CB 1110 ? "OFF"
JC 1120 POSITION 2,16:?" 
PRESS X TO EXIT BACK
TO EDITOR"
JA 1130 CURX=38:CURY=2:POKE
764,255:OLY=CURY:CLO
SE #1:OPEN #1,4,0,"K
:"
LF 1140 REM RUN THE MENU
FI 1150 POSITION 0,CURY:?">
":POSITION 38,CURY:?"
"<"
HB 1160 IF PEEK(764)<>255 TH
EN 1200
NL 1170 IF STRIG(0)=0 THEN 1
570
NL 1180 GOTO 1160
PA 1190 REM TOP PART (BITS 0
-7)
FJ 1200 IF PEEK(764)<>255 TH
EN GET #1,K
GF 1210 IF K=88 THEN ? "
{CLEAR}":GOTO 400:RE
M TO VOICE DEVELOPIN
G SCREEN
NK 1220 IF CURY<10 AND (K=48
OR K=49) THEN 1260
ND 1230 IF K=65 OR K=90 THEN
1270
NE 1240 IF K=32 AND CURY>9 T
HEN 1310

```



```

JK 1250 POKE 764,255:GOTO 11
50
NB 1260 BIT(CURY-2)=K-48:POS
ITION CURX-1,CURY: ?
BIT(CURY-2):POKE 764
,255:GOTO 1150
KM 1270 CURY=CURY-1*(K=65):C
URY=CURY+1*(K=70):IF
CURY<2 THEN CURY=14
CB 1280 IF CURY>14 THEN CURY
=2
LA 1290 POSITION CURX,OLY: ?
CHR$(32):POSITION 0,
OLY: ? CHR$(32):OLY=C
URY
ME 1300 GOTO 1150
OO 1310 POKE 752,0
OL 1320 TRAP 1320:POSITION 1
,18: ? "DEL LINE": ;
INPUT NUM:TRAP 40000
AF 1330 IF NUM<>INT(ABS(NUM)
) OR NUM<0 THEN 1320
OO 1340 ON CURY-9 GOTO 1360,
1390,1420,1460,1490
AO 1350 REM PICK A CHANNEL
BF 1360 IF NUM>3 THEN POSITI
ON 1,18: ? "
DEL LINE)Channel (0
-3)":FOR D=1 TO 200:
NEXT D:GOTO 1320
EM 1370 CHAN=NUM:GOTO 1510
PA 1380 REM PICK A VOLUME
DN 1390 IF NUM>15 THEN POSIT
ION 1,18: ? "
DEL LINE)Volume (0-
15)":FOR D=1 TO 200:
NEXT D:GOTO 1320
DM 1400 VO(CHAN)=NUM:GOTO 15
10
CB 1410 REM PICK A DISTORTIO
N
OM 1420 IF NUM>14 THEN POSIT
ION 1,18: ? "
DEL LINE)Distortion
(0-14)":FOR D=1 TO
200:NEXT D:GOTO 1320
AK 1430 IF NUM/2<>INT(NUM/2)
THEN NUM=NUM-1:IF N
UM<0 THEN NUM=0
CA 1440 SD(CHAN,1)=NUM*16:GO
TO 1510
NI 1450 REM PICK A FREQUENCY
OB 1460 IF NUM>255 THEN POSI
TION 1,18: ? "
DEL LINE)Frequency
(0-255)":FOR D=1 TO
200:NEXT D:GOTO 1320
JB 1470 SD(CHAN,0)=NUM:GOTO
1510
PN 1480 REM PICK A STATUS
CB 1490 IF NUM>1 THEN POSITI
ON 1,18: ? "
DEL LINE)Status (0=
OFF 1=ON)":FOR D=1 T
O 300:NEXT D:GOTO 13
20
ME 1500 STAT(CHAN)=NUM:GOTO
1510
CP 1510 POSITION 1,18: ? "
(5 DEL LINE)":POKE 7
52,1
BN 1520 POSITION 14,10: ? CHA
N:POSITION 14,11: ? V
O(CHAN): ? "
BJ 1530 POSITION 14,12: ? SD(
CHAN,1)/16: ? " : POSI
TION 14,13: ? SD(CHAN
,0): ? "
KB 1540 POSITION 14,14: IF ST
AT(CHAN)=1 THEN ? "O
N ":GOTO 1560
CJ 1550 ? "OFF"
NM 1560 GOTO 1150

DC 1570 REM TURN ON ALL VOIC
E CHANNELS
BP 1580 BYTE=0
OO 1590 IF BIT(0) THEN BYTE=
BYTE+1
OI 1600 IF BIT(1) THEN BYTE=
BYTE+2
ON 1610 IF BIT(2) THEN BYTE=
BYTE+4
PC 1620 IF BIT(3) THEN BYTE=
BYTE+8
CO 1630 IF BIT(4) THEN BYTE=
BYTE+16
CD 1640 IF BIT(5) THEN BYTE=
BYTE+32
CK 1650 IF BIT(6) THEN BYTE=
BYTE+64
FN 1660 IF BIT(7) THEN BYTE=
BYTE+128
DM 1670 POKE 53768,0:POKE 53
768,BYTE
GD 1680 FOR X=0 TO 3:IF STAT
(X)=0 THEN 1700
NN 1690 POKE 53760+(X*2),SD(
X,0):POKE 53761+(X*2
),SD(X,1)+VO(X)
FP 1700 NEXT X
DC 1710 POSITION 0,1: ? "PRE
SS TRIGGER TO TURN O
FF ALL VOICES"
MK 1720 IF STRIG(0)=1 THEN 1
720
BC 1730 POSITION 0,1: ? "PRE
SS TRIGGER TO LISTEN
TO ALL VOICES"
KL 1740 FOR X=0 TO 8:POKE 53
760+X,0:NEXT X:GOTO
1150
EA 1750 REM CLEAR BARS
ID 1760 COLOR 124:FOR A=35 T
O 1 STEP -1:R=S(CH,A
):X=A+2:Y=(20-R):PLO
T X,Y
EP 1770 NEXT A
JL 1780 COLOR 32:PLOT OLD,21
ME 1790 POSITION 13,1: ? "
NB 1800 POSITION 16,3: ? "
(3 SPACES)"
JM 1810 GOTO 180
LD 1820 REM LISTEN MENU
MD 1830 POKE 710,0:COLOR 1:P
OKE 89,SCREEN1:POKE
106,SCREEN1+4:POKE D
L+5,SCREEN1: ? CHR$(1
25):POKE 752,1: ? "
(CLEAR)"
HE 1840 POSITION 0,0: ? "(Q)
(38 R)(E)":FOR D=1 T
O 14:POSITION 0,D: ?
"|" : POSITION 39,D: ?
"|"
EI 1850 NEXT D:POSITION 0,15
: ? "(Z)(38 R)(C)"
AL 1860 POSITION 8,2: ? "
(4 SPACES)PLAY VOICE
S MENU(4 SPACES)"
DP 1870 POSITION 7,4: ? " --
Voice numbers --"
KN 1880 POSITION 6,6: ? "A -
0,1,2,3(9 SPACES)B -
0,1,2"
PH 1890 POSITION 6,7: ? "C -
0,1,3(11 SPACES)D - 0
,1"
PB 1900 POSITION 6,8: ? "E -
0,2,3(11 SPACES)F - 0
,2"
PD 1910 POSITION 6,9: ? "G -
0,3(13 SPACES)H - 1,2
,3"
MN 1920 POSITION 6,10: ? "I -
1,2(13 SPACES)J - 1,
3"

HF 1930 POSITION 6,11: ? "K -
2,3(13 SPACES)L - 0"
BN 1940 POSITION 6,12: ? "M -
1(15 SPACES)N - 2"
IA 1950 POSITION 6,13: ? "O -
3(15 SPACES)P - PRE
SENT
MENU"
GC 1960 POSITION 8,15: ? "
(W)(20 R)(W)"
CJ 1970 POSITION 8,16: ? "I Yo
ur choice (A-P) : "
FF 1980 POSITION 8,17: ? "
(Z)(20 R)(C)"
DG 1990 GET #1,K:IF K<65 OR
K>80 THEN POSITION 8
,16: ? "I YOUR CHOICE
(A-P) : " :GOTO 1970
KH 2000 K=K-64:IF K=16 THEN
180
HM 2010 POKE 53768,BYTE:POKE
53775,3:POKE 53760,
SD(0,0):POKE 53762,S
D(1,0):POKE 53764,SD
(2,0):POKE 53766,SD(
3,0)
MN 2020 POKE 559,0:FOR A=1 T
O 35
IB 2030 POKE 53761,(SD(0,1)+
S(0,A))*G(K,1)=1
IH 2040 POKE 53763,(SD(1,1)+
S(1,A))*G(K,2)=1
IN 2050 POKE 53765,(SD(2,1)+
S(2,A))*G(K,3)=1
JD 2060 POKE 53767,(SD(3,1)+
S(3,A))*G(K,4)=1
OI 2070 NEXT A:POKE 559,34
JK 2080 POKE 53768,0:POKE 53
761,0:POKE 53763,0:P
OKE 53765,0:POKE 537
67,0
JN 2090 POSITION 14,16: ? "CH
OICE : ?":GOTO 1970
DH 2100 REM LOAD/SAVE/DIR/DE
L
HD 2110 POKE 89,SCREEN1:POKE
106,SCREEN1+4:POKE
DL+5,SCREEN1
AA 2120 CLOSE #1:OPEN #1,4,0
,"K:"
GH 2130 GRAPHICS 0:POKE 710,
0:POKE 82,0:POKE 752
,1:TRAP 40000:POKE 7
12,192
IM 2140 ? "(CLEAR) INPUT/OUT
PUT MENU"
ID 2150 ? "1 - Directory of
*.SND files"
KL 2160 ? "2 - Save envelope
s file"
JO 2170 ? "3 - Load envelope
s file"
AA 2180 ? "4 - Delete envelo
pe file"
AL 2190 ? "5 - MAIN MENU"
NM 2200 POSITION 0,15: ? "CHO
ICE : ?"
EE 2210 GET #1,K:POSITION 9,
15: ? CHR$(K):IF K<49
OR K>53 THEN 2130
BB 2220 IF K=53 THEN GOTO 40
DM 2230 ON K-48 GOSUB 2260,2
350,2830,2530,180
DD 2240 REM DIR,SAVE,LOAD,DE
L
MI 2250 GOTO 2130
CM 2260 ? "(CLEAR) DIRECTORY
OF DRIVE 1":POKE 82,
2: ? :POKE 712,4
CJ 2270 CLOSE #2:OPEN #2,6,0
,"D1:*.SND":POKE 82,
2:FLN=0
FO 2280 TRAP 2310:INPUT #2,F
L$:FLN=FLN+1:IF FL$(

```



```

5,8)="FREE" THEN 231
0
BA 2290 FLL$=FL$(3,10):FLL$(
9,9)="":FLL$(10,12)
=FL$(11,13)
EF 2300 ? FLL$:GOTO 2280
JJ 2310 POKE 82,0: ? : ? FL$:C
LOSE #2:POKE 82,2
IC 2320 ? : ? "Press RETURN t
o continue..."
JD 2330 GET #1,K:IF K<>155 T
HEN 2330
KJ 2340 RETURN
EO 2350 ? "(CLEAR) SAVE A *
SND FILE":POKE 712,
80
FP 2360 ? : ? : ? :POKE 752,0
AD 2370 ? "Enter name for fi
le."
FE 2380 ? " or X to exit."
PD 2390 ? "{3 SPACES}{Q}{22
R}{E}"
NL 2400 ? "{3 SPACES}{Dn:fil
ename. Extender!"
PJ 2410 ? "{3 SPACES}{automa
tically attached!"
PE 2420 ? "{3 SPACES}{Z}{22
R}{C}"
HL 2430 GOSUB 2630:IF FL$="X
" THEN RETURN
EA 2440 IF PEEK(195)=170 THE
N 2480
BH 2450 ? : ? FL$:" already e
xists.": ? "Do you wa
nt to rewrite it (Y/
N) ?"
BH 2460 GET #1,K:IF K<>ASC("
Y") AND K<>ASC("N")
THEN 2460
JH 2470 IF K=ASC("N") THEN 2
350
DH 2480 ? "Okay, saving ";FL
$;". "
OF 2490 CLOSE #2:OPEN #2,8,0
,FL$:PUT #2,BYTE
CA 2500 FOR X=0 TO 3:FOR Y=0
TO 1:PUT #2,SD(X,Y)
:NEXT Y:NEXT X
BF 2510 FOR X=0 TO 3:FOR Y=1
TO 35:PUT #2,S(X,Y)
:NEXT Y:NEXT X
KJ 2520 RETURN
NE 2530 ? "(CLEAR) DELETE A *
SND FILE":POKE 71
2,64
FP 2540 ? : ? : ? :POKE 752,0
HB 2550 ? "Enter file to del
ete."
FE 2560 ? " or X to exit."
PD 2570 ? "{3 SPACES}{Q}{22
R}{E}"
OE 2580 ? "{3 SPACES}{Dn:fil
ename. Extender!"
AC 2590 ? "{3 SPACES}{automa
tically attached!"
PE 2600 ? "{3 SPACES}{Z}{22
R}{C}"
HL 2610 GOSUB 2630:IF FL$="X
" THEN RETURN
MO 2620 GOTO 2720
PE 2630 POKE 82,13
OB 2640 POSITION 12,14: ? "
{DEL LINE}":TRAP 26
40:INPUT FL$:TRAP 40
000:IF FL$="" THEN 2
640
PK 2650 IF FL$="X" THEN RETU
RN
FD 2660 IF FL$(1,1)<>"D" AND
FL$(3,3)<>"": THEN
2640
OO 2670 POKE 82,0: ? :TRAP 26
90:FOR D=4 TO 15:IF

```

```

FL$(D,D)="." THEN PO
P :GOTO 2640
NEXT D
KO 2690 FL$(LEN(FL$)+1)="."SN
D"
GC 2700 CLOSE #2:TRAP 2710:0
PEN #2,4,0,FL$:POKE
82,0:CLOSE #2
KK 2710 RETURN
HO 2720 IF PEEK(195)<>170 TH
EN 2750
FD 2730 ? "Sorry, ";FL$;" do
es not": ? "
(5 SPACES)seem to ex
ist. Please try aga
in...":TRAP 40000
BE 2740 POKE 752,1:POSITION
12,20: ? " Press any
key":GET #1,K:GOTO
2530
BN 2750 ? "Attention ";FL$;
" will": ? "be utterl
y destroyed after de
letion."
LD 2760 REM
KA 2770 ? : ? "Press c to con
tinue or RETURN to a
sort"
OM 2780 GET #1,K:IF K<>ASC("
C") AND K<>155 THEN
2780
HM 2790 IF K=155 THEN 2530
BD 2800 ? CHR$(125): ? : ? "NO
W DELETING ";FL$;".
"
DM 2810 XIO 33,#2,0,0,FL$
KM 2820 RETURN
HC 2830 ? "(CLEAR) LOAD A *
SND FILE":POKE 712,
242

```

Program 2: Sound Player

```

EA 10 GRAPHICS 0:POKE 710,12
8:POKE 712,8:POKE 82,0
LF 20 DIM SD(3,1),S(3,35),FL
$(20):OPEN #1,4,0,"K:"
:POKE 752,1
BC 30 POSITION 0,0: ? "
(8 SPACES)SOUND DEVELO
PMENT SYSTEM
(8 SPACES)"
BJ 40 POSITION 0,1: ? "
(14 SPACES)SOUND PLAYER
(14 SPACES)"
AI 50 ? : ? "This program loa
ds and plays sound
(6 SPACES)envelopes sa
ved with the Sound Edi
tor."
EB 60 ? : ? "It can also be u
sed as a routine in yo
rown programs."
CF 80 POSITION 3,22: ? "----
--- Press any key ---
-----"
OB 90 POKE 764,255:GET #1,K
AG 100 ? CHR$(125):POSITION
0,0: ? "(8 SPACES)SOUN
D DEVELOPMENT SYSTEM
(8 SPACES)":POKE 752,
0
FL 110 ? : ? : ? "Enter name f
or load file."
OO 120 ? "{Q}{25 R}{E}"
IF 130 ? "Dn:filename. Ext
ender is!"
PI 140 ? "I automatically app
ended. I"
PI 150 ? "{Z}{25 R}{C}"
EB 160 POSITION 12,14:INPUT
FL$:IF FL$="" THEN 16
0

```

```

HO 170 IF FL$(1,1)<>"D" THEN
100
EB 180 POKE 82,0: ? :TRAP 200
:FOR D=1 TO 15:IF FL$
(D,D)="." THEN POP :G
OTO 160
BN 190 NEXT D
GP 200 FL$(LEN(FL$)+1)="."SND
"
MH 210 CLOSE #2:TRAP 220:OPE
N #2,4,0,FL$:CLOSE #2
AP 220 IF PEEK(195)<>170 THE
N 240
IK 230 ? : ? FL$:" does not s
eem to exist...":POKE
752,1:POSITION 0,20:
? "{6 SPACES}PRESS AN
Y KEY":GOTO 90
FF 240 ? "Okay, loading ";FL
$;". "
JA 250 CLOSE #2:OPEN #2,4,0,
FL$:GET #2,BYTE
AD 260 FOR X=0 TO 3:FOR Y=0
TO 1:GET #2,Z:SD(X,Y)
=Z:NEXT Y:NEXT X
PI 270 FOR X=0 TO 3:FOR Y=1
TO 35:GET #2,Z:S(X,Y)
=Z:NEXT Y:NEXT X
LP 280 REM ENVELOPE PLAYED H
ERE
MJ 290 POKE 559,0
EM 300 POKE 53768,BYTE:POKE
53775,3:POKE 53760,SD
(0,0):POKE 53762,SD(1
,0):POKE 53764,SD(2,0
):POKE 53766,SD(3,0)
JN 310 POKE 559,0:FOR A=1 TO
35
CI 320 POKE 53761,(SD(0,1)+S
(0,A))
CN 330 POKE 53763,(SD(1,1)+S
(1,A))
DC 340 POKE 53765,(SD(2,1)+S
(2,A))
DH 350 POKE 53767,(SD(3,1)+S
(3,A))
CA 360 NEXT A:FOR D=1 TO 5:N
EXT D:POKE 559,34
PM 370 POKE 53761,0:POKE 537
63,0:POKE 53765,0:POK
E 53767,0:POKE 752,1
EI 380 POSITION 0,21: ? "Pres
s : SPACE BAR to hear
again": ? "
(8 SPACES)RETURN to 1
oad another"
DI 390 GET #1,K:IF K<>32 AND
K<>155 THEN 390
NK 400 IF K=32 THEN 290
EA 410 IF K=155 THEN GOTO 10
0

```

Program 3: Sound Program Writer

```

ED 10 GRAPHICS 0:POKE 710,0:
POKE 82,0:POKE 752,1:0
PEN #1,4,0,"K:"
BB 20 POSITION 0,0: ? "
(8 SPACES)SOUND DEVELO
PMENT SYSTEM
(8 SPACES)"
OD 30 POSITION 10,1: ? "SOUND
PROGRAM MAKER":GOSUB
1040
AH 40 ? : ? : ? "This program
takes any arrangement
of(3 SPACES)envelopes
made with the Sound Ed
itor"
IE 50 ? "and writes an ENTER
-able BASIC program t
o play them. All you n

```



```

eed to provide"
KN 60 ? "is the starting line
number of the
(6 SPACES)program and
a saved sound envelope
file."
KJ 70 POSITION 13,20: ? "PRESS
ANY KEY"
EI 80 POKE 764,255:GET #1,K:
? CHR$(125)
FK 100 ? : ? : ? "Enter name f
or load file."
DN 110 ? "{Q}{25 R}{E}"
IE 120 ? "IDn:filename. Ext
ender is!"
PH 130 ? "I automatically app
ended. !"
PH 140 ? "{Z}{25 R}{C}"
HK 150 POKE 752,0:POSITION 1
2,14:INPUT FL$:IF FL$
=" " THEN 90
FF 160 IF FL$(1,1)<>"D" THEN
90
BK 170 POKE 82,0: ? :TRAP 190
:FOR D=1 TO 15:IF FL$
(D,D)="." THEN POP :G
OTO 90
BM 180 NEXT D
HH 190 FL$(LEN(FL$)+1)="."SND
"
HF 200 CLOSE #2:TRAP 210:OPE
N #2,4,0,FL$:CLOSE #2
HC 210 TRAP 4000:IF PEEK(19
5)<>170 THEN 230
II 220 ? : ? FL$: "does not s
eem to exist...":POKE
752,1:POSITION 0,20:
? "{6 SPACES}PRESS AN
FE 230 ? "Okay, loading ";FL
$: " "
IP 240 CLOSE #2:OPEN #2,4,0,
FL$:GET #2,BYTE
AC 250 FOR X=0 TO 3:FOR Y=0
TO 1:GET #2,Z:SD(X,Y)
=Z:NEXT Y:NEXT X
PH 260 FOR X=0 TO 3:FOR Y=1
TO 35:GET #2,Z:S(X,Y)
=Z:NEXT Y:NEXT X
EH 270 POKE 752,1: ? CHR$(125
): ? : ? "The next m
enu is a listing of d
ifferent voice arrang
ements. Choose one,"
BN 280 ? "enter the name of
the program to write,
and then select the
starting line"
HM 290 ? "number; the comput
er will start writing
"
PL 300 ? "the program, which
should only take a
(3 SPACES)few minutes
." : POSITION 13,20
DN 310 ? "PRESS ANY KEY":POK
E 764,255:GET #1,K
LO 320 ? CHR$(125)
PD 340 POSITION 12,2: ? "ARRA
NGEMENT MENU"
BB 350 POSITION 10,4: ? " -- V
oice Numbers --"
HF 360 POSITION 6,6: ? "A - 0
,1,2,3(9 SPACES)B - 0
,1,2"
LP 370 POSITION 6,7: ? "C - 0
,1,3(11 SPACES)D - 0,1
"
MH 380 POSITION 6,8: ? "E - 0
,2,3(11 SPACES)F - 0,2
"
MP 390 POSITION 6,9: ? "G - 0
,3(13 SPACES)H - 1,2,3
"
JF 400 POSITION 6,10: ? "I -
1,2(13 SPACES)J - 1,3"
DN 410 POSITION 6,11: ? "K -
2,3(13 SPACES)L - 0"
OF 420 POSITION 6,12: ? "M -
1(15 SPACES)N - 2"
DO 430 POSITION 6,13: ? "O -
3"
DI 440 POSITION 9,16: ? "Your
choice (A-P) : ?"
NE 450 GET #1,K:IF K<65 OR K
>79 THEN POSITION 9,1
6: ? "YOUR CHOICE (A-P
): " : GOTO 440
AP 460 V=K-64
OI 480 ? CHR$(125): ? : ? "Ent
er name for the progr
am file."
OA 490 POKE 752,0:POSITION 1
2,14: ? "{DEL LINE}";:
INPUT FLL$:IF FLL$=" "
THEN 480
NC 500 IF FLL$(1,1)<>"D" THE
N 480
KL 520 ? CHR$(125): ? : ? "Ent
er the starting line
number": ? "and interv
al for ";FLL$
DI 530 ? " Enter START,INTER
VAL"
JB 540 TRAP 540:POSITION 0,1
2:INPUT N1,N2:TRAP 40
000
JH 550 ? "OK...": ? " --COMPAC
TING--"
EP 560 LN=35:FOR A=35 TO 1 S
TEP -1
JN 570 S=0:FOR B=1 TO 4:IF G
(V,B)=0 THEN GOTO 590
FN 580 S=S+(B-1,A)
AF 590 NEXT B:NUM=N1
EL 600 IF S<>0 THEN POP :GOT
O 620
GJ 610 LN=A:NEXT A
AN 620 ? " --WRAPPING--"
BB 630 CLOSE #1:OPEN #1,8,0,
FLL$
KB 640 D$=STR$(N1):D$(LEN(D$
)+1)=" " : LN=D$(LEN(D$
)+1)=STR$(LN*2): ? #1;
D$:D$=" " : N1=N1+N2
LD 650 D$=STR$(N1):D$(LEN(D$
)+1)=" " : POKE 53775,3:P
OKE 53768,0: ? #1;D$:
D$=" " : N1=N1+N2
HA 660 D$=STR$(N1):D$(LEN(D$
)+1)=" " : DIM "
KH 670 FOR A=1 TO 4:IF G(V,A
)<>1 THEN 690
OF 680 D$(LEN(D$)+1)="E":D$(
LEN(D$)+1)=STR$(A-1):
D$(LEN(D$)+1)="$(LN),
KL 690 NEXT A:IF D$(LEN(D$),
LEN(D$))="," THEN D$(
LEN(D$))=" "
BA 700 ? #1;D$:N1=N1+N2:D$="
"
PK 710 FOR A=0 TO 3:IF G(V,A
+1)<>1 THEN 760
FB 720 D$=STR$(N1):D$(LEN(D$
)+1)=" " : E=D$(LEN(D$)+
1)=STR$(A):D$(LEN(D$)+
1)="$( " : D$(LEN(D$)+1
)=CHR$(34)
HP 730 FOR B=1 TO LN:R=S(A,B
):IF R<10 THEN D$(LEN
(D$)+1)="0"
NN 740 D$(LEN(D$)+1)=STR$(R)
:NEXT B:D$(LEN(D$)+1)
=CHR$(34)
BF 750 ? #1;D$:D$=" " : N1=N1+N
2
BN 760 NEXT A
HJ 770 D$=STR$(N1):D$(LEN(D$
)+1)=" " : POKE 53768," :D
$(LEN(D$)+1)=STR$(BYT
E): ? #1;D$:N1=N1+N2:D
$=" "
DB 780 D$=STR$(N1)
PP 790 FOR A=0 TO 3:IF G(V,A
+1)<>1 THEN 820
PP 800 D$(LEN(D$)+1)=" " : POKE
":D$(LEN(D$)+1)=STR$(
53760+2*A):D$(LEN(D$
)+1)=" " : D$(LEN(D$)+1)
=STR$(SD(A,0))
AB 810 D$(LEN(D$)+1)=" " :
LE 820 NEXT A:IF D$(LEN(D$),
LEN(D$))="," THEN D$(
LEN(D$))=" "
BE 830 ? #1;D$:N1=N1+N2:D$="
"
CO 840 D$=STR$(N1)
CB 850 D$(LEN(D$)+1)=" " : POKE
559,0:FOR A=1 TO 1: ? D$
(LEN(D$)+1)=STR$(LN):
D$(LEN(D$)+1)=" " : STEP
2: ? #1;D$:D$=" " : N1=N
1+N2
PN 860 FOR A=0 TO 3:IF G(V,A
+1)<>1 THEN 910
DB 870 D$=STR$(N1)
BE 880 D$(LEN(D$)+1)=" " : POKE
":D$(LEN(D$)+1)=STR$(
53761+2*A):D$(LEN(D$
)+1)=" " : VAL(E):D$(LEN(D
$)+1)=STR$(A)
HE 890 D$(LEN(D$)+1)="$(A*2-
1,A*2): ? D$(LEN(D$)+
1)=STR$(SD(A,1))
OP 900 ? #1;D$:N1=N1+N2
BK 910 NEXT A
BH 920 D$=STR$(N1):D$(LEN(D$
)+1)=" " : NEXT A:POKE 55
9,34: ? #1;D$:D$=" " : N
1=N1+N2
OH 930 D$=STR$(N1):D$(LEN(D$
)+1)=" " : POKE 53761,0:P
OKE 53763,0:POKE 5376
5,0:POKE 53767,0:POKE
53768,0:END: ? #1;D$:
PK 940 D$=" " : ? #1;D$:CLOSE #
1
HA 950 ? CHR$(125): ? : ? " --
> FINISHED":POKE 752,
1
BE 960 ? : ? : ? "Press [E] to r
un the program again"
KJ 970 ? "{3 SPACES}or [E] to
quit."
KL 990 OPEN #1,4,0,"K":POKE
764,255
PN 1000 GET #1,K:IF K<>ASC("
R") AND K<>ASC("Q")
THEN 1000
LL 1010 IF K=ASC("R") THEN R
UN
PN 1020 POKE 752,0:END
CF 1040 DIM FL$(20),G(14,4),
FLL$(20),S(3,35),D$(
120),SD(3,1)
NM 1050 RESTORE 1060:FOR X=1
TO 14:FOR Y=1 TO 4:
READ D:G(X,Y)=D:NEXT
Y:NEXT X
ID 1060 DATA 1,1,1,1,1,1,0
,1,1,0,1,1,1,0,0,1,0
,1,1,1,0,1,0,1,0,0,1
,0,1,1,1,0,1,1,0,0,1
,0,1,0,0,1,1,1,0,0,0
,0,1,0,0
JI 1070 DATA 0,0,1,0,0,0,0,1
KJ 1080 RETURN

```

©

IBM Keyboard Customizer

David Engebretsen

This tutorial for the IBM PC/PCjr explains how to customize your computer's keyboard with simple DOS 2.0 or higher commands. Besides reassigning key definitions to your own personal taste, you can create keyboard macros, define as many as 40 function keys and choose new screen modes and color combinations.

Have you ever wished you could change a key on the keyboard into another key, or make a single keystroke spell out an entire phrase? Would you like to move the colon and the semicolon keys, or put the Return key in a more convenient position? It is quite possible to do all this and more with a few simple commands from DOS. You can even create up to 40 different function keys which can be used to print a variety of command words or phrases of any length. With a little more work, you can even change your standard QWERTY keyboard into the efficient Dvorak format which can improve typing speed dramatically.

All this is made possible with the extended screen and keyboard control offered by DOS 2.0 and higher. Though the DOS manual devotes only one page to this subject, the process is not complicated. Let's look first at how to switch key assignments. Then we'll explore how to perform related tasks such as setting the screen mode, changing the background and foreground colors, and positioning the cursor.

Boot Up With CONFIG.SYS

It is surprisingly easy to reassign any key or keys to a location that suits your own personal needs. The first step is to create a CONFIG.SYS file that installs an extended screen and keyboard control device driver when you boot the system. This can be done with the EDLIN system editor included on your DOS disk. From the DOS command prompt (>) simply type EDLIN CONFIG.SYS and press Return. The drive will whir as it opens a new file; after a few moments your screen should look something like this:

New file

*

Type the following lines, pressing Return at the end of each line:

```
11  
DEVICE=ANSISYS *
```

Press Ctrl-Break, then type the number 3 and press Return. At this point you have created a configuration file that runs automatically whenever you boot the computer. The result is that the computer is then made ready to accept some new key assignments.

Reassigning Key Definitions

The next step is to do the actual key switching. Since this can involve some odd character sequences, it's easiest to do this from within a BASIC program that stores the needed data in a text file on disk. Here is a program that demon-

strates the technique. We'll use it to create a file that changes the uppercase Q to an uppercase D.

```
DN 10 A$=CHR$(27) + "[" + CHR$(34) + "Q" + CHR$(34) + ";" +  
+ CHR$(34) + "D" + CHR$(34) + "p"  
FN 20 OPEN "KEY.TXT" FOR OUTPUT  
AS #1  
DD 30 PRINT #1,A$  
GN 40 CLOSE #1
```

Save this program as REASSIGN.BAS and run it. REASSIGN.BAS creates a text file that contains the following character sequence:

ESC ["Q"; "D"p

CHR\$(27) is the ASCII code for the ESC character; this is the control code which changes the uppercase Q into an uppercase D. To implement this change, insert the disk containing your new CONFIG.SYS file, then reboot by pressing Ctrl-Alt-Del simultaneously. This enables the ANSI device driver which in turn allows the keyboard to be redefined.

After answering the time and date prompts, type TYPE KEY.TXT at the DOS prompt and press Return. This action enters the special control characters into the computer's memory. Now whenever you type an uppercase Q, the system substitutes an uppercase D.

Keyboard Macros

The same technique can be used to create a *keyboard macro*—a key that produces a multicharacter word or phrase with just one keystroke. To illustrate, let's redefine uppercase Q so that it prints the phrase *The*

Phrase whenever it's pressed. Reenter BASIC and load the REAS-SIGN.BAS program again, then replace line 10 as shown here:

```
EP 10 A$=CHR$(27)+"["+CHR$(34)+"
    Q"+CHR$(34)+" ";" +CHR$(34)+"
    The Phrase"+CHR$(34)+"p"
```

Now run the program again. This creates a text file that contains these characters:

```
ESC ["Q";"The Phrase"p
```

Type SYSTEM to go back to DOS, then type the KEY.TXT file again. Now whenever you press Q the computer prints *The Phrase* on the screen.

When creating the KEY.TXT file, it is also acceptable to substitute an ASCII code for the character. For example, say that you want to change uppercase Q back to uppercase D. Go back to BASIC again and change line 10 as shown here. Note that instead of D we are using 68, the ASCII code for D:

```
PH 10 A$=CHR$(27)+"[B1;68p"
```

Run the program, enter DOS, and TYPE the program. Uppercase Q should again produce a D.

40 Function Keys

Now let's create some extra function keys. By supplying an extended ASCII code, you can redefine the ten function keys alone or in conjunction with the Ctrl, Shift, or Alt keys. That comes to four sets of ten, or 40 keys. Rerun the example program after changing line 10 as shown here:

```
AG 10 A$=CHR$(27)+"[0;84;" +CHR$(
    34)+"DIR"+CHR$(34)+" ";" 13p"
```

The following text file is created:

```
ESC [0;84;"DIR";p
```

The 0 before the 84 tells the computer to look for an *extended key-code*—a code that signals a special key combination. The extended keycode 84 represents Shift-F1. What we've done is redefine this key combination so that it prints DIR followed by a carriage return. Run the program, exit to DOS, and type KEY.TXT again. Hold down Shift and press F1: the disk directory is displayed.

Note the number 13 just before the p in this character sequence. This is the ASCII code for Return. Adding this character to the end of

a character sequence has the same effect as pressing RETURN manually on the keyboard. The computer types the letters D-I-R, then issues a Return to carry out the command. You can find a complete list of all the extended keycodes on page G-7 of the IBM BASIC manual.

Screen Modes And Colors

Using a similar method, you can also change the screen color or shift to a different screen resolution. To change colors, replace the lowercase p in line 10 with a lowercase m, and supply an appropriate color number. For instance, change line 10 as shown here and create a new KEY.TXT file:

```
EN 10 A$=CHR$(27)+"[37;44m"
```

Now the program creates this text file:

```
ESC [37;44m
```

When this file is TYPED from DOS the screen turns blue.

The same procedure works for changing the screen mode. Change line 10 to this (note that an h is substituted for the m in the preceding example):

```
BS 10 A$=CHR$(27)+"[1h"
```

When you TYPE the resulting file from DOS, the screen goes into 40 × 25 color text mode. To obtain 320 × 200 color graphics mode, simply change the number 1 in line 10 to a 4. Pages 13-9 and 13-10 in the DOS 2.0 manual contain a complete listing of all the numbers for different screen modes and color combinations.

The customizations you create using these techniques will stay in effect as long as you are working in DOS or a DOS-related program such as DEBUG or EDLIN. If you're tired of the normal white-on-black screen display, this simple technique can bring a welcome change. Note, however, that these changes disappear if you reboot the computer, go to BASIC, or run an application that imposes its own definitions on the system. ©

Copies of articles from this publication are now available from the UMI Article Clearinghouse.

For more information about the Clearinghouse, please fill out and mail back the coupon below.

UMI Article Clearinghouse

Yes! I would like to know more about UMI Article Clearinghouse. I am interested in electronic ordering through the following system(s):

- ☐ DIALOG/Dialorder ☐ ITT Dialcom
☐ OnTyme ☐ OCLC ILL
 Subsystem

- ☐ Other (please specify) _____
☐ I am interested in sending my order by mail.
☐ Please send me your current catalog and user instructions for the system(s) I checked above.

Name _____
 Title _____
 Institution/Company _____
 Department _____
 Address _____
 City _____ State _____ Zip _____
 Phone (____) _____

Mail to: University Microfilms International
 300 North Zeeb Road, Box 91 Ann Arbor, MI 48106

Advanced Programming On The Atari ST

Writing sophisticated programs on the Atari ST requires a more thorough understanding of the computer's operating system than was necessary on earlier machines. This article is an introduction to the various operating system routines available to ST programmers. It is an excerpt from COMPUTE!'s ST Programmers Guide (by the editors of COMPUTE! Publications.)

It seems quite natural to move the mouse pointer to an icon, click on it, and then double-click to open the window. Often you can choose options from menus at the top of the screen by simply pointing and clicking. Click on the menu bar and you can move the window around. You can resize it, expand it to fill the screen, or make the window go away, all with a few clicks of the mouse.

Making things simple for the user requires a wide variety of fairly complex routines for handling input, output, graphics, and so on. These routines are invisible to the user, who simply moves the mouse to point and click. But as a programmer, you may find it helpful to gain some acquaintance with the built-in TOS and GEM routines. The more you know about how they work, the more control you will have over the machine and the more power you can put into programs.

You can call many of these routines in BASIC with the GEMSYS or VDISYS commands, but to get the maximum speed and power from your ST, you'll need either a C compiler or a machine

language assembler.

Alphabet Soup

We'll be referring to the various collections of routines by their initials: TOS, VDI, AES, and so on. We'll first look at what they are and what they do.

The Operating System (TOS) is either built into your ST computer in ROM or on a TOS boot disk. The most identifiable element of this operating system is the *Graphics Environment Manager* (GEM) desktop. However, the operation of TOS involves the interplay of many different, specialized components. The first step toward understanding GEM and TOS is to learn the name and function of each of the parts.

The desktop environment is actually a special type of GEM application, which exists only to perform file operations, including running other GEM applications. Every function that it performs, from examining disk directories to running other applications, can be duplicated by any GEM application using the facilities of the AES, VDI, and GEMDOS.

The *GEM Virtual Device Interface* (VDI) provides low-level graphics display and mouse input routines. This routine library includes primitive drawing operations like line, marker, circle, and polygon, as well as display management routines like clipping and block image copying.

The *GEM Application Environment Services*, AES for short, performs higher-level graphic and data management operations for maintaining the GEM desktop environ-

ment. Built on top of the GEMDOS file system and the GEM VDI, the AES routines make it much easier for programs to perform mouse and window operations.

The *Disk Operating System* GEMDOS performs character-oriented file and device input/output (I/O). Many of its routines are used by the GEM AES. GEM applications can also use GEMDOS for file access and for device operations. For the actual low-level operations, GEMDOS calls the *Basic Input/Output System* (BIOS), a group of routines which perform machine-specific tasks on the ST. The Atari XBIOS provides additional machine-specific operations. They are not used by GEMDOS, but are available to applications which need routines not available through GEMDOS or the GEM BIOS.

AES In Detail

When you double-click on an application's icon, the desktop starts executing that application. Although you can move or resize an application's window, its output appears only in that window. An application can redraw its window when it is partially covered by an accessory without overwriting the accessory's window. When you click on the box at the upper left of an application's window, the application stops executing and the desktop returns.

All of these operations would be much harder to perform without the support of the AES, which is composed of a process dispatcher, a screen manager, a desk accessory buffer, and 11 subroutine libraries.

The process dispatcher allows one application and several accessories to wait for a user action simultaneously, a limited form of multitasking. When the system is booted, the accessory buffer is loaded with accessories. The process dispatcher suspends all accessories until a menu item for one of them is selected.

The procedures in the subroutine libraries support common operations on the application environment, the desktop. An application could perform most desktop operations without the AES, using the same VDI routines, but the standard AES routines make the job considerably less difficult. This is a powerful motivation for programmers to make user interfaces more alike, so the programs are easier to learn and use. Nonetheless, nonstandard interfaces can be built where needed, using either VDI or lower-level AES routines.

While the information presented here is not a thorough treatment of the AES, it is intended to be a concise overview of the whole library. It should give you some idea of how the AES is organized, what its constituent parts are and how they interact, and what facilities are available to the GEM programmer. More specific details about each routine's parameters and their values can be found in the Digital Research GEM literature and in sample GEM programs.

Application Library

The application library is, in effect, the gateway to the rest of the AES. Its `appl_init` routine is used by an application to register with the AES and obtain a process ID. It also tells the AES to set up data structures to keep track of this application or accessory. When the program has finished, the `appl_exit` routine tells the AES to deallocate the ID number and data structures for this application.

When waiting for user input from the mouse or keyboard, a program can call one of the routines in the event library instead of waiting in a loop. Besides saving the programmer from writing a few more lines of code, a call to the event library freezes the application and allows possible multitasking, if an

event for another frozen process occurs first. Calls are available to wait for keyboard, mouse button, message, or timer events. Messages indicating that some action must be performed—like redrawing or moving a window—are usually awaited with an event library call. The routine most commonly used is `evnt_multi`, which allows an application to wait for more than one event at once.

The `menu_bar` routine, in the menu library, is used to display or erase the menu bar. Another routine, `menu_register`, is called by desk accessories to add a name to the Desk menu. Routines are also available to enable and disable menu items, to change the names of menu items, and to display a menu item in reverse video (as a selected item).

Windows

The single most distinguishing feature of the GEM operating environment is the window. The output of applications and accessories running on the ST is displayed on the screen in separate windows, many of which can be moved or changed in size with the mouse. There can be as many as eight windows on the screen at once, and they can overlap in any way, but applications are advised not to use more than four of them if the Desk menu is displayed. Since desk accessories need to have windows available when they are activated, it's best to reserve four of the available windows for accessories. The routines in the window library perform services which are useful in the management of these windows.

The `wind_create` and `wind_delete` routines are used to create windows and to dispose of them. When an application calls `wind_create` to generate a new window, it establishes the maximum possible size of the window and the features with which the window is endowed, such as sliders, a name bar, a full screen box, and so forth. This routine returns a numeric identifier for the new window, called a *handle*. The `wind_open` call actually displays the window on the screen at a particular location. Conversely, a window can be hidden with the `wind_close` routine. The expanding

and shrinking box effects that are seen when many applications open and close windows are not part of the window library subroutines. Rather, they are independent effects routines from the graphics library, which an application can call for a little more flash.

Several window library routines provide information about windows. `Wind_get` can provide information about windows, including a window's position and size, and the positions of its vertical and horizontal sliders if it is so equipped. It can also return the handle of the top window on the screen (the window with the highest priority), as well as the set of rectangles that make up the visible portion of a window's work area, which can be an irregular shape if a window is partially covered by another. The `wind_set` routine is used to change the size and position of a window, the positions of its sliders or its name, or the set of controls attached to the window. It can also move one window to the top of the list of windows.

To determine which window the mouse currently points to, the `wind_find` routine can be called. `Wind_calc` doesn't operate on any particular window, but instead determines the work area size of a hypothetical window, given its external size and the set of controls it contains. It can also perform the reverse calculation, determining the external dimensions of a window.

The `wind_update` routine tells AES that an application is going to draw in a window or that an update is finished. When a window is being updated, no alerts, dialogs, or menus will be displayed in front of the window.

Objects

Most items on the GEM desktop, including menus, alerts, and even windows, are organized as object trees. GEM *objects* include icons, strings, graphic boxes, and editable text fields. *Trees*—linked lists—of these objects can be managed with the routines in the object library. This library includes routines to add and delete objects from a tree, to compare the mouse's position to that of an object, to let the user edit a text object, and to draw the entire

tree on the screen. Object trees are usually stored in a file separate from the application that uses them. These *resource files* can be handled with the resource library, described below.

A *form* is a standard mechanism for getting information from the user. A form usually includes at least one modifiable object, like a text string or a button. In the case of a *dialog*, one type of form, the program needs to call the `form_dial` routine to indicate that a dialog is beginning, and then draw the object tree which comprises the dialog, using the `obj_draw` routine from the object library. The `form_do` routine should then be called to perform the interaction. Another call to `form_dial` restores the area of the screen where the dialog took place.

Two other forms, the *alert* and the *error box*, are more limited in their content and, accordingly, easier to implement. In the case of an alert, the AES builds an object tree containing the text string passed in the call to `form_alert` and handles all the details of display and interaction during that one call. The error box is even simpler. All that is needed is the number of a GEMDOS error code. An object tree containing the text which corresponds with that error will be displayed when `form_error` is called.

Special-Purpose Libraries

The AES graphics library routines are lower-level interfaces to the display screen and mouse. The most commonly used routine in this library is `graf_handle`, which returns the handle—the identifier—for the currently opened VDI workstation. Every GEM application must call this routine at its start, after calling `appl_init`, since the handle is needed to open a new VDI virtual workstation to draw in. It is also necessary for calling VDI drawing routines.

To manage the mouse at a low level, the `graf_mouse` routine can be used to change the shape of the cursor. `graf_mkstate` monitors the positions of the mouse, its buttons, and the keyboard, while `graf_watchbox` modifies the state of a box object depending on whether or not the mouse's pointer is inside

or outside the box.

Several common graphic effects are also available through the graphics library. `Graf_rubberbox` draws a rectangle between a fixed point and the mouse's position, changing the size of the box as the mouse moves. A moving box of fixed size attached to the mouse's position can be animated with `graf_drawbox`, while `graf_movebox` just moves a box between two positions without any consideration of the mouse. `Graf_growbox` and `graf_shrinkbox` are two animation routines which can be called when opening and closing windows, respectively, to show a box which moves and changes size.

The file selector library contains but one lonely routine, `fsel_input`. This displays a standard dialog box, called a *file selector*, on the screen and allows the user to choose a filename from the directories of the various disks in the system. When the interaction is complete, it returns the pathname of the selected file to the calling application.

The objects that an application uses—its menus, dialogs, and so forth—can be stored separately from the application's code in a resource file. Resource files containing these objects are created with the *GEM Resource Construction Set* program. The resource library can then be used to load this file and to access its contents.

The `rsrc_load` routine searches for a resource file with a particular name and attempts to load it into memory. `Rsrc_gaddr` can be used by the application to find the address of a particular object or tree in the resource file that has been loaded. To allow applications to run with different screen resolutions in different display modes, all the sizes and positions of objects in a resource file can be expressed in characters instead of pixels. When the resource file is loaded, `rsrc_obfix` must be called to convert the sizes into pixels in the current display mode. When a resource file is no longer needed, `rsrc_free` deallocates the memory space that the resource occupies.

Using GEMDOS

For many kinds of programs, the

windowed GEM environment might not be needed, so an alternative is available. A complete set of character-oriented I/O functions is provided in the GEMDOS library. Unlike many operating systems, GEMDOS is easily called from languages like C. Instead of taking parameters in the microprocessor's internal registers, which are not directly controllable in high-level languages, parameters are passed to these routines on the stack in the same way that parameters are passed between C functions.

Even GEM applications need to call GEMDOS at least occasionally. For instance, the AES scrap facility expects applications to store and read the contents of the Clipboard directly from disk. Any program which handles some kind of document—a spreadsheet, word processor, database, and so on—will also need to call GEMDOS to load and store files.

To call GEMDOS, the 68000 microprocessor's TRAP #1 instruction must be executed. The last word pushed on the stack gives the number of the routine requested. If the routine returns a value, it will be stored in the 68000's D0 register. Although this register cannot be directly read by a C program, C functions also return results in this register. A GEMDOS call can return a value in exactly the same manner as a call to another C function.

Process Functions

- 00 `Pterm0()`. Terminate with a return code of 0.
- 49 `Ptermres(size,code)`. Terminate, but keep the program's code in memory. A 32-bit parameter indicates how much memory should remain allocated. A 16-bit parameter gives the return code. This function is used by background programs, like print spoolers, to give up control of the foreground process.
- 75 `Pexec(runflag,pathname,tail,envi-ron)`. Load a program from disk. A one-word parameter indicates whether it should be run or not (00=run, 03=load only). The second parameter is a pointer to the pathname of the file. Parameter 3 points to a command tail for the program, and parameter 4 is a pointer to its environment strings. If the file was loaded only, the result of the function is the load address. If it was executed, the result is its return code.
- 76 `Pterm(code)`. Terminate, returning a one-word return code.

Device I/O Functions

- 01 **Cconin()**. Read a character from the console. No parameters are needed.
- 02 **Cconout(char)**. Write a character to the console. A single, word-length parameter contains the character in its low byte.
- 03 **Cauxin()**. Read a character from the auxiliary device.
- 04 **Cauxout(char)**. Write a character to the auxiliary device.
- 05 **Cprnout(char)**. Write a character to the printer.
- 06 **Crawio(char)**. If the parameter is not \$00FF, write it as a character to the console; otherwise, return a character from the console with no echo, including control characters.
- 07 **Crawcin()**. Read a character from the console with no echo and no control character trapping.
- 08 **Cnecin()**. Read a character from the console with no echo, but trap ^C, ^S, and ^Q.
- 09 **Cconws(string)**. Write a zero-terminated string to the console. The long word parameter contains the address of the string.
- 10 **Cconrs(buffer)**. Input a line of characters from the console, allowing line editing. The long word parameter contains the address of a buffer, the first byte of which holds the buffer's length. The second byte of the buffer will get the length of the string, and the string will be zero-terminated.
- 11 **Cconis()**. Check the status of the console input device. Returns -1 if a character is waiting, 0 if none is available.
- 16 **Cconos()**. Check the status of the console output device. Returns -1 if it is ready to receive a character, 0 if it is not ready.
- 17 **Cprnos()**. Check the status of the print device.
- 18 **Cauxis()**. Check the status of the auxiliary input device.
- 19 **Cauxos()**. Check the status of the auxiliary output device.

Time Functions

- 42 **Tgetdate()**. Return the current system date. Bits 0-4 of the result contain the date, 5-8 contain the month, 9-15 contain the year minus 1980 (up to 2099).
- 43 **Tsetdate(date)**. Set the current system date to the word value in the parameter.
- 44 **Tgettime()**. Return the current system time. Bits 0-4 contain seconds/2, 5-10 contain minutes, 11-15 contain hours.
- 45 **Tsettime(time)**. Set the current system time to the word value in the parameter.

System Functions

- 32 **Super()**. Enter 68000 supervisor mode.
- 48 **Sversion()**. Return the GEM version number.

Drive Functions

- 14 **Dsetdrv(drive)**. Set the default disk drive to the value passed as a parameter. Values 0-15 indicate drives A-P.
- 25 **Dgetdrv()**. Return the value of the current default drive.
- 54 **Dfree(buffer,drive)**. Ask for information about a disk. The first parameter contains the address of buffer to receive the information. The second parameter is a 16-bit value indicating the drive being queried. The buffer gets four values: free space, number of clusters on drive, size of sector in bytes, size of cluster in sectors.
- 57 **Dcreate(pathname)**. Create a subdirectory. The long word parameter contains the address of null-terminated string for the pathname of the new directory. If the result is non-zero, the operation failed.
- 58 **Ddelete(pathname)**. Delete a subdirectory.
- 59 **Dsetpath(pathname)**. Set the current default pathname to the string addressed by the parameter.
- 71 **Dgetpath(buffer,drive)**. Store the current directory for a drive in a 64-byte buffer addressed by the first parameter. The second parameter indicates which drive (00=default drive, 01-10=A-P).

File Functions

- 26 **Fsetdta(DTAbuffer)**. Use the long word parameter to set the address for a 44-byte file data buffer. This buffer is used only when searching for a file (routines 78 and 79).
- 47 **Fgetdta()**. Return the address of the file data buffer.
- 60 **Fcreate(pathname,attributes)**. Create a file named by the string addressed by the first parameter. An additional 16-bit parameter indicates the file's attributes (01=RO, 02=hidden). The result is a 16-bit file handle.
- 61 **Fopen(pathname,access)**. Open the file named by the string addressed by the first parameter. An additional 16-bit parameter indicates type of access (00=read only, 01=write only, 02=read and write). The handle of the file is returned as the function's result.
- 62 **Fclose(handle)**. Close a file. The handle of the file is passed as a parameter.
- 63 **Fread(handle,count,buffer)**. Read from a file. The first parameter is the handle of an open file and the second is a four-byte count for the transfer. The third parameter is the address of a buffer to which the bytes are to be read. The result returned by the function is the number of bytes.
- 64 **Fwrite(handle,count,buffer)**. Write to a file. The function uses the same parameters as Fread.
- 65 **Fdelete(pathname)**. Delete the file named by the string pointed to by the parameter.

- 66 **Fseek(count,handle,operation)**. Move the file pointer. The first parameter is a signed long word representing a byte count. The second parameter is a file handle. The third indicates the meaning of the byte count (00=absolute position *N* bytes after the start of file, 01=*N* bytes forward or backward from current position, 02=*N* bytes before the end of the file). As a result, it returns the absolute file pointer position.
- 67 **Fattrib(pathname,operation,attributes)**. Read or change the attributes of file. The first parameter is a pointer to a pathname for the file. The second parameter is 00 for get, 01 for set. The third parameter is a word containing the attributes.
- 69 **Fdup(handle)**. Return a copy of the file handle passed as a parameter.
- 70 **Fforce(handle1,handle2)**. Force the first parameter, a file handle, to point to same device as the second parameter, also a handle.
- 78 **Fsfirst(pathname,attributes)**. Search for the first file which matches the search string addressed by parameter 1. The string can contain the * and ? wildcards. Parameter 2 contains the attribute flags of the file. The 44-byte file data buffer set by Fsetdta holds size of file in bytes 26-29, and the file's name and type in bytes 30-43.
- 79 **Fsnext()**. Search for another match to the file, using the data buffer at DTA. This function takes no parameters.
- 86 **Frename(0,oldname,newname)**. Rename a file. Parameter 1 is a word with the value 0; parameter 2 is a pointer to the old pathname of the file; parameter 3 points to the new pathname.
- 87 **Fdtime(buffer,handle,operation)**. Get or set a file's date and time information. Parameter 1 is a pointer to a two-word buffer—a time word and a date word. Parameter 2 is a file handle, and parameter 3 is a word which indicates which operation to perform (00=set, 01=get).

Memory Functions

- 72 **Malloc(count)**. Allocate some number of bytes to the calling application. The length of the block requested is passed in the parameter, a long word. It returns a value of 0 if the request fails, or the address of the block allocated if it succeeds. If the parameter has a value of -1, the number of free bytes is returned instead.
- 73 **Mfree(address)**. Free a block of memory. The four-byte parameter should contain the address of the block.
- 74 **Mshrink(0,address,length)**. Reduce the size of an allocated block of memory. The first parameter must be a word with value 0, the second parameter is the address of the block, and the third parameter is the number of bytes which should remain in the block.

Block PEEK And POKE For Atari

Ronald R. Lambert

Here is a convenient way to eliminate long initialization delays caused by POKEing large amounts of data into memory. It works entirely in BASIC and works very fast. The demonstration program moves the entire character set in an instant and redefines the keyboard as a Dvorak layout. This technique can be used on all Atari 400/800, XL, and XE computers and is recommended for intermediate to advanced BASIC programmers.

PEEK and POKE are among the fastest commands in BASIC. But because they handle only one byte at a time, it can take a while to transfer large blocks of data from one area of memory to another. We've all waited while programs with long loops PEEK a series of memory locations, or READ numbers from DATA statements, and then POKE the numbers into memory somewhere else. Perhaps the program is redefining the character set, or setting up player/missile graphics, or building a machine language subroutine, or creating a new keyboard definition table. Whatever's happening, it slows things down.

Lengthy FOR-NEXT loops with PEEK and POKE or READ and POKE are the primary cause for tedious delays while these programs initialize. No one likes to sit staring at a blank screen for very long. The program usually prints a

message like "Please wait while I initialize," but isn't there a better way? Sometimes a machine language subroutine can help speed things up, but if you can't write in ML yourself, finding a routine exactly suited to the needs of your program can be difficult.

Fortunately, Atari BASIC's flexibility provides a solution. It's possible to transfer large blocks of data from Read Only Memory (ROM) or program lines to any area of Random Access Memory (RAM) virtually instantaneously—with BASIC commands only. The secret is called the *string offset* technique. By modifying the variable value table (a section of memory which keeps track of BASIC program variables), you can redefine any string and relocate it anywhere in memory.

Here's a quick overview of how the technique works. Suppose you set up a string called ROM\$ which contains a block of data found in ROM—the character set data, for instance. Next, you set up another string called RAM\$ in the area of RAM to which you want to move the data contained in ROM\$. To copy the data from ROM to RAM, then, all that's required is the simple statement `RAM$=ROM$`. Is that easy enough? Using the string offset technique, any portion of ROM—or all of ROM, if you make the strings big enough—can be copied into RAM in the blink of an eye.

The Variable Value Table

To use the string offset technique, you have to learn how to modify the variable value table and the string offset pointers. This isn't too difficult if you tackle the job one step at a time.

The first step is to make things easier for ourselves by insuring that ROM\$ and RAM\$ are the first variables found in the table. We can do this by making ROM\$ and RAM\$ the very first variables defined in the program. Enter NEW as a direct command before typing the first program line containing these names. Then dimension the variables in this order:

```
10 DIM ROM$(length),RAM$(length)
```

where *length* is the length of each string in characters as required by your program. If you're moving character set data, for instance, you'd dimension these strings to the number of bytes in the character set—1024 bytes in graphics mode 0 or 512 bytes in modes 1 and 2. (Usually you'll dimension ROM\$ and RAM\$ to the same length if you're transferring a block of memory.)

However, if you are using an Atari 400 or 800 with the old BASIC revision A, a major caveat applies. *You cannot move blocks of memory that are exact multiples of 256 bytes.* Attempting to move blocks of this size will trigger the infamous BASIC lockup bug, freezing your

computer until you turn the power off and back on—which will, of course, result in the loss of your program. [For more information on the lockup bug, see this month's "Readers' Feedback" column.-Ed] You can determine your version of BASIC by entering PRINT PEEK (43234). If the value returned is 162, you have revision A. If 96 is returned, you have revision B (built into most 600XL and 800XL models), and 234 indicates revision C, available on cartridge from Atari and built into the XE models.

The string offset technique will work as described with revision A as long as you make sure your block length is not an exact multiple of 256. So for this example you should substitute 1025 instead of 1024. This will transfer an extra byte of memory following the character set, but that doesn't cause any problems and it prevents the lock-up bug from biting.

The second step is to make BASIC think that ROM\$ is actually 1024 bytes long (remember, use 1025 for revision A BASIC). The DIM statement reserves memory for the string but doesn't actually define the string. Use a line like this:

```
20 ROM$(length)=""
```

By defining the last character in the string as a space, BASIC is forced to treat ROM\$ as a 1024-character-long string, even though no other characters have been defined.

The third step is to calculate the location of the variable value table in memory, with a statement like this:

```
30 VT=PEEK(134)+PEEK(135)*256
```

The variable VT equals the starting location of the variable value table. Each string which is declared in an Atari BASIC program has eight bytes in this table. We'll see the significance of these bytes in a moment.

After these variables are set up, the first eight bytes in the variable value table (VT to VT+7) contain information for ROM\$, and the next eight bytes (VT+8 to VT+15) contain information for RAM\$.

Locating The Strings

To use the string offset technique, we're primarily interested in the

third and fourth bytes for each of these two variables in the variable value table. The memory locations for these bytes can be expressed as VT+2 and VT+3 for ROM\$, and VT+10 and VT+11 for RAM\$.

What do these bytes signify? Briefly, each pair of bytes is a low-byte/high-byte combination that indicates the *relative displacement* of each string from the starting location of the first string in the program. Since we've made sure that the first string in the program is ROM\$, the values stored in VT+2 and VT+3 for ROM\$ will both be zero. And since we've also made sure that RAM\$ is the second string in the program, the values stored in VT+10 and VT+11 for RAM\$ depend on the length of ROM\$.

For instance, if ROM\$ is dimensioned to 1024, then the memory which BASIC sets aside for RAM\$ must begin 1024 bytes after the start of ROM\$ to leave room for ROM\$. Therefore, the value stored in VT+10 is zero, and the value stored in VT+11 is four. (Since VT+11 is the high byte of the offset, it's multiplied by 256, which equals 1024.)

Actually, the memory for ROM\$ and RAM\$ does not begin at these locations. Instead, you have to add another value indicated by the *string offset pointers* at memory locations 140 and 141. If you use this statement:

```
40 SF=PEEK(140)+PEEK(141)*256
```

then the variable SF returns the number that should be added to the relative displacement values given in the variable value table. (Since the relative displacement of the first string is zero, this means that SF always equals the address of the first string.)

The reason for this seemingly complicated arrangement, incidentally, is that the computer can now easily relocate strings as the program length changes simply by altering the offset pointers.

Setting The Table

Now it's clear how the string offset technique works: We can relocate a string anywhere in memory by merely POKEing different values into its relative displacement indicators in the variable value table.

For example, suppose we want

to move ROM\$ to the starting memory address of the standard character set in ROM, which is location 57344. We subtract the amount of the string offset (SF) from 57344, and convert the remainder into low-byte/high-byte numbers. Then all we have to do is POKE LS into VT+2 and POKE HS into VT+3, and ROM\$ is moved to the proper location. The statements might look like this:

```
50 S=57344-SF:HS=INT(S/256):
```

```
LS=S-HS*256
```

```
60 POKE VT+2,LS:POKE VT+3,HS
```

Now we can turn our attention to RAM\$.

The usual place to set up a new character set is below RAMTOP—the memory location returned by PEEK(106)*256. Some people prefer to move RAMTOP down by POKEing a lower number into register 106, issue a new GRAPHICS command to set up a new display list and screen memory below the altered RAMTOP, and then put the new character set above the new RAMTOP. There are advantages and disadvantages to each method, including a "RAMTOP dragon" to watch out for. We'll stick to the easiest method for this example. Let's simply put the new character set eight pages (2048 bytes) below RAMTOP. This leaves enough room for the 1024-byte character set, plus another 1024 bytes for the display list and screen memory in graphics mode 0.

We move RAM\$ to this location by figuring the proper values and POKEing them into VT+10 and VT+11:

```
70 RAMPAGE=PEEK(106)-8
```

```
80 S=RAMPAGE*256-SF:HS=INT
```

```
(S/256):LS=S-HS*256
```

```
90 POKE VT+10,LS:POKE VT+11,HS
```

Finally, all that remains is one simple step:

```
100 RAM$=ROM$
```

Instantly, the character set in ROM is copied into RAM, where it can be customized to suit our purposes.

Two Potential Problems

There are two things to watch out for when using the string offset technique. First, if you set up a string in a section of RAM where vital tables or pointers are stored,

then do anything to change the contents of the string, or press BREAK and enter a direct command (which causes BASIC to shift strings and all their contents in memory), the computer may behave very strangely. You'll probably have to turn the machine off and on again to regain control.

Second, you cannot POKE a negative number into the variable value table without getting an error message. How, then, can you move a string to a location in memory lower than the value (SF) indicated by the offset pointers? Simple. POKE the offset pointers to zero, and POKE memory locations 140 and 141 to zero. (Make sure you do this before relocating any strings, or they'll all be moved again when you change the offset pointers.) *But don't leave zeros in locations 140 and 141.* Save the original values and POKE them back in when you're finished transferring the data.

A Dvorak Demo

The program following this article demonstrates the string offset technique and accomplishes several things. First, it copies the standard character set from ROM into RAM (eight pages below RAMTOP) and modifies it so that the CTRL key characters can be recognized more easily. If you press CTRL-A, for example, you won't get the usual graphics symbol; you'll get an underlined A, so you can see at a

glance which keys to press to type that character. This way, you can enter ATASCII (Atari ASCII) characters directly into memory from statements in program lines without using DATA statements and slow, one-byte-at-a-time POKES, because all the characters are immediately recognizable. This character set modification is accomplished in an eye-blink.

Second, the program copies the keyboard definition table from ROM and loads it into memory page 6, a normally unused portion of RAM from locations 1536-1791. Then the program modifies the keyboard table to create a Dvorak keyboard layout. Designed by August Dvorak after 20 years of scientific study and testing, the Dvorak keyboard makes things as easy as possible for typists, in contrast to the conventional QWERTY keyboard, which doesn't put the most frequently used keys on the home row. Many typists are able to convert from QWERTY to Dvorak touch-typing within a few days, and often find they can type faster with substantially fewer errors.

The Dvorak keyboard portion of the program will not work with the older 400 and 800 models because it relies on the KEYDEF pointer at locations 121-122. This pointer was added to the improved operating system in the XL and XE models, and is not implemented in the original Atari operating system

ROMs. Owners of 400s and 800s can still use the redefined character set portion of the example by simply omitting all lines numbered higher than 215. If you are using revision A BASIC, you'll also need to change the 1024s in lines 10 and 20 to 1025s.

Notice that this program must deal with the problem mentioned above: The memory address 1536 is lower than the value for SF, so the string offset pointers at locations 140 and 141 have to be changed.

A FOR-NEXT loop is used to enter ATASCII characters 0 through 26, so this part of the program takes a little longer—almost a whole second. You could make it run even faster by typing the CTRL key characters directly in string assignment statements, as seen in lines 140 to 170. This is where the new character set could come in handy.

As a final bonus, the program demonstrates a customized keyboard entry routine that works faster than the GET function. It does this by reading a hardware location (53769), then using the keyboard conversion table located in ROM (64337 to 64592) to translate the keyboard codes into ATASCII codes the same way the operating system does.

When the program runs, it lets you toggle back and forth from QWERTY to Dvorak, just like on an Apple IIc. Press SHIFT-ESC to

Atari Dvorak Keyboard Layout

| | | | | | | | | | | | | | | | | |
|---------|---|---|---|----|---|---|---|---|---|---|-------|---|-------|--------|--------|--------|
| ESC | ! | @ | # | \$ | % | ^ | & | \ | [| (|] |) | CLEAR | INSERT | DELETE | BREAK |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | | | < | > | BACK | |
| | | | | | | | | | | | | | | | SPACE | |
| TAB | ? | , | . | P | Y | F | G | C | R | L | ↑ | ↓ | ↑ | ↓ | + | RETURN |
| | / | , | . | | | | | | | | | | | | = | |
| CONTROL | A | O | E | U | I | D | H | T | N | S | ← | → | ← | → | * | CAPS |
| | | | | | | | | | | | | | | | | |
| SHIFT | : | Q | J | K | X | B | M | W | V | Z | SHIFT | ■ | | | | |
| | ; | | | | | | | | | | | | | | | |

(800 XL keyboard shown; others may vary slightly)

toggle. If you become a real Dvorak fan, you can even find keycap stickers at many office supply stores to modify your keyboard. The accompanying figure shows the Dvorak layout.

Additional Notes

A few modifications to the standard Dvorak layout were necessary because of the special functions and extra keys on the Atari keyboard. The seldom-used brackets may be typed with CTRL-9 or CTRL-0. The += key, normally located at the upper right of the Dvorak keyboard, has been moved down. The * \ key has been retained in its standard Atari position because these characters have extra use as arithmetic functions in programming. Since the Atari has no cent symbol, this has been replaced with the vertical line as uppercase 6. In place of the asterisk (uppercase 8 on the Dvorak keyboard) is the backslash. The ' " key has been exchanged with the ; : key to avoid conflict between the CTRL-up arrow and CTRL-semicolon.

The Atari logo key is the inverse video key on the Atari 400/800, and it is reversed with the right SHIFT key on those models. Regrettably, the Atari has no dash in its character set. While the useless underline could be redefined as a dash for screen display, most Atari printers also lack the dash.

If you enter NEW or load a new program after this one is run, the new character set with readable CTRL key characters remains active (as long as the new character set is not overwritten). Press SYSTEM RESET or POKE 756,224 to restore the old character set. The following POKES switch on the Dvorak keyboard even after a NEW command: POKE 121,0:POKE 122,6. To switch back to QWERTY, use POKE 121,81:POKE 122,251.

The next time you need to transfer large blocks of data from one portion of memory to another, try using the string offset technique. It gives you the best of both worlds: the convenience of BASIC and near-machine language speed. Never again will you have to sit staring at a blank screen waiting for your programs to move large amounts of data in memory.

Dvorak Keyboard Demo

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!.

```
CH 10 DIM ROM$(1024),RAM$(1024):REM These variable
names must be the first entered.
HI 20 ROM$(1024)=" ":GOSUB 60:S=57344-SF:GOSUB 70:
POKE VT+2,LS:POKE VT+3,HS:REM This moves ROM $.
GH 30 RAMPAGE=PEEK(106)-8:S=RAMPAGE*256-SF:GOSUB 70:
POKE VT+10,LS:POKE VT+11,HS:REM This moves RAM$.
IF 40 RAM$=ROM$:REM That is all it takes! ROM data is now copied into RAM.
PD 50 SETCOLOR 1,0,0:SETCOLOR 2,0,10:SETCOLOR 4,8,6:GOTO 80
IP 60 VT=PEEK(134)+PEEK(135)*256:SF=PEEK(140)+PEEK(141)*256:RETURN
KH 70 HS=INT(S/256):LS=S-HS*256:RETURN
IA 75 REM Now modify the character set:
KH 80 RAM$(513,520)=RAM$(98):RAM$(521,728)=RAM$(26,6):FOR X=520 TO 728 STEP 8:RAM$(X,X)=CHR$(255):NEXT X
OK 90 RAM$(769,776)=RAM$(114):RAM$(776,776)=CHR$(255):RAM$(985,992)=RAM$(218):RAM$(992,992)=CHR$(255)
MP 100 POKE 756,RAMPAGE:REM Set the CHBAS pointer to start of new character set.
ON 110 ? "New character set ready.":? "CONTROL A=";CHR$(1);", CONTROL Z=";CHR$(26)
MD 115 REM Press RESET or POKE 756,224 to restore old character set.
AB 120 CLR:DIM ROM$(256),RAM$(256):ROM$(256)=" ":GOSUB 60:LSF=PEEK(140):HSF=PEEK(141):POKE 140,0:POKE 141,0
DE 125 REM Now copy keyboard definition table from ROM to page 6 of RAM:
NH 130 S=64337:GOSUB 70:POKE VT+2,LS:POKE VT+3,HS:S=1536:GOSUB 70:POKE VT+10,LS:POKE VT+11,HS:RAM$=ROM$
LL 135 REM Now change keyboard definition table to Dvorak layout:
IH 140 RAM$="nhs":RAM$(6)="t-r lg":RAM$(14)="c'=k j":RAM$(22)="xq":RAM$(33)="w vb mz":RAM$(41)="p .f"
BJ 150 RAM$(46)="y, /":RAM$(57)="ude":RAM$(62)="io":RAM$(65)="NHS":RAM$(70)="T ^R LG":RAM$(78)="C +K J/ XQ:"
NI 160 POKE 1614,34:RAM$(92)="!":RAM$(95)="@!W VB
```

```
MZ":RAM$(105)="P .F":RAM$(110)="Y, ? ( ) &"
CH 170 RAM$(118)="\" :RAM$(121)="UDE":RAM$(126)="I O"
FK 180 FOR X=0 TO 26:READ A:RAM$(A)=CHR$(X):NEXT X:POKE 1687,123:POKE 1706,96:RAM$(177)="[" :RAM$(179)="]"
GH 190 DATA 175,192,164,142,186,187,172,140,130,190,147,145,139,166
JC 200 DATA 129,191,169,151,137,131,134,185,163,161,150,174,167
HA 205 REM Now restore offset pointers and set up custom keyboard entry routine:
LL 210 POKE 140,LSF:POKE 141,HSF:CLR:DIM K$(1):QD=64337:LK=9:C=0:I=0:
DB 215 REM Press SHIFT ESC to toggle between QWERTY and Dvorak keyboard.
CE 220 ON PEEK(753)<>3 GOTO 220:K=PEEK(53769):IF K=39 OR K=60 OR K=92 THEN GOTO K*10
NS 230 A=PEEK(QD+K):K=K+C*(A>96 AND A<123):K$=CHR$(PEEK(QD+K)+I):? K$:POKE 753,3*(LK=K):POKE 20,0
JA 240 ON PEEK(753)<3 GOTO 260:IF PEEK(20)<24 THEN N 240
MN 250 IF PEEK(753)=3 THEN ? K$:GOTO 250
JA 260 LK=K:POKE 764,255:GOTO 220
JE 265 REM Type a letter twice then hold it down to start autorepeat.
IL 390 I=128*(I=0):GOTO 610:REM Inverse video toggle
FD 600 C=64*(C=0):REM Cap/lowcase toggle
MH 610 POKE 753,0:GOTO 260
ES 920 QD=1536+62801*(QD=1536):POKE 712,134+66*(QD=1536):GOTO 610:REM Green border=Dvorak. Blue=QWERTY.
```

©

Attention Programmers

COMPUTE! magazine is currently looking for quality articles on Commodore, Atari, Apple, and IBM computers (including the Commodore Amiga and Atari ST). If you have an interesting home application, educational program, programming utility, or game, submit it to COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Or write for a copy of our "Writer's Guidelines."

Screen Machine II: A Sketchpad With Pull-Down Menus For PC and PCjr Part 1

Charles Brannon, Program Editor

Pull-down menus in IBM BASIC? It's no fantasy—presented here is a full-featured drawing program that illustrates the convenience and flexibility of a menu-driven user interface. Next month, in Part 2, we'll show how you can add these menu routines to your own programs. This month's drawing program, "Screen Machine II," runs on any IBM PC or compatible with a color/graphics adapter and BASICA, or a PCjr with Cartridge BASIC. Joystick or touch tablet optional.

Software features first popularized by the Apple Macintosh are finding wider acceptance throughout the computer industry. Pull-down menus and point-and-click selections have become a way of life among owners of the Macintosh, Commodore Amiga, and Atari ST. With the advent of operating system veneers such as *Microsoft Windows*, *GEM*, and *Topview* for IBM machines, even more people are getting excited about mousing around on their computer.

Without the software tools to implement these techniques, though, programmers have to laboriously write all the routines needed for pull-down menus, icon selection, and windowing, taking time away from programming the application itself. Tools such as *Windows* and *GEM* do the trick for advanced programmers, but BASIC programmers have to reinvent the wheel if they want to add these useful features.

You might think BASIC is not fast enough to emulate the features of operating systems written in turbocharged 8088 or 68000 machine language, but it's almost always possible to tease just a little more power out of BASIC. Although a complete mouse-based user interface is a bit much to expect, I've developed a set of generalized subroutines that any BASIC programmer can use to support fancy pull-down menus in *Microsoft Advanced BASIC (BASICA)* or *PCjr Cartridge BASIC*. The routines require bit-mapped graphics, so you need a color/graphics adapter if you're using a PC or compatible. (The PCjr has a built-in color/gra-

phics adapter.) By changing only a few small subroutines, the package can be adapted for other graphics cards and pointing devices.

Rather than illustrate these routines with a plain-vanilla demo program, I thought I'd provide a more convincing illustration: a full-featured drawing sketchpad. "Screen Machine II" is a descendant of the original "Screen Machine," a drawing program published about two years ago in *COMPUTE!'s PC & PCjr Magazine*.

The original Screen Machine used a traditional command-driven user interface. Individual keystrokes were required to activate special commands. For example, to draw a line, you first pressed the space bar to "nail down" one endpoint, marking the spot with a cross. You then moved the cursor to a new position and then pressed L to connect the marked spot with the new cursor position. A line was drawn to connect the points. To draw a circle, you first set your mark to represent the center of the circle, then moved the cursor to a point along the desired circumference. You had to visualize the circle

in your mind, because it wasn't actually drawn until you pressed C.

Although Screen Machine had plenty of features (and in the hands of a talented artist was capable of making beautiful pictures), the stumbling block was the indirect method of using the program. You had to memorize every command or frequently refer to a list of commands. This approach works well once you've mastered a program, but it can alienate the newcomer or occasional user. In a drawing program, especially, it's crucial not to break the flow of ideas between the artist and the canvas.

Screen Machine II

Thanks to pull-down menus, you don't have to memorize a lot of commands to have fun with Screen Machine II. All of the functions are available for selection whenever you need them.

The listing following this article is the minimum required to publish Screen Machine II as a ready-to-run program. Screen Machine II needs almost all of BASIC's 64K memory space, so the original program listing with full comments didn't leave enough memory to run. Next month, however, we'll present a fully REMarked version that shows exactly how the program works, along with a tutorial on using the menu routines in your own programs.

When you first run Screen Machine II, there is a short delay, then the drawing screen appears. The top line of the screen shows which menus are available: *Picture*, *Tools*, and *Preferences*. Your color palette appears at the bottom of the screen, initially showing boxes filled with cyan, magenta, and white paint.

A pointer cursor appears near the middle of the screen. You use this pointer to select items from menus, dip into the paint to change your drawing color, or to draw figures. The pointer can be controlled with a joystick, a touch tablet such as the KoalaPad, or the cursor keys (make sure Num Lock is in the correct position for cursor control). When using the joystick, you may want to unlock it for free-floating movement.

If you don't have a joystick or touch tablet, you can disable the

joystick routine in Screen Machine II to prevent interference with the cursor controls. Change line 340 from `FROZEN=0` to `FROZEN=-1`.

To use a light pen or mouse controller, you need to modify the subroutine at line 20000, which we'll discuss next month. The use of menus, though, is not tied to the actual pointing device used, such as a mouse.

Calibrate Your Joystick

The pointer responds most naturally with a touch tablet, so the program is initially set up to use a KoalaPad. The KoalaPad simulates the joystick, but has a greater range, so if you use a joystick with Screen Machine II you'll only be able to position the pointer within the upper-left quadrant of the screen. You may also have problems using a different touch tablet, since not all tablets return the same values. When you first run the program, then, you need to *calibrate* your joystick or tablet. The calibration option is available under the *Preferences* menu, but we haven't discussed how to use the menus yet. And until you calibrate your joystick, you probably can't access the menu item that is used to select the calibrate option. Fortunately, you can also press the J key—a keyboard shortcut—to activate the calibration feature.

After you press J, you're first asked to move the joystick to the upper-left corner, then press the button. (Screen Machine II only uses the top button on the joystick, or the left button of the touch tablet.) This first action sets the origin of your pointing device. If you're using a touch tablet, it is vital that at this point you merely lift the pen off the tablet surface and press the button. This lets Screen Machine II know when you are pressing down on the tablet, and when you lift the pen off the tablet. The value for "pen up" is the same as the coordinates for the upper-left position of the tablet, so it's best just to press the button without touching the tablet surface, in order to make sure that Calibrate sees the right value. With a joystick, move the stick to the northwest corner before you press the button.

Next, you're asked to move the

joystick to the lower-right corner, then press the button. With the touch tablet, move the stylus or your fingertip to the southeast corner of the tablet, and while pressing *firmly* with the stylus, click the button. It may be best to use a position slightly above and to the left of the lower-right position, since if you put the pen off the tablet surface, no value is generated.

In general, you must press very firmly against the tablet surface, almost to the point of scoring the tablet, in order to avoid false readings caused by intermittent stylus contact. These false readings aren't caused by Screen Machine II, but by the tablet. A special routine could be used to compute the average position of the touch tablet within the last second, then reject values far out of range, but this would slow down the program to a crawl. As it turns out, this jitter is rarely a problem, since the BASIC program samples the touch-tablet too slowly to see many of the transient glitches.

After calibration, you should be able to move the pointer freely as you slide your finger or stylus across the tablet surface, or by moving the joystick. Control may seem clumsy at first, especially with the joystick, but improves considerably with practice. If you get no response at all, check the joystick or tablet cabling, and press J to calibrate again.

Keyboard Control

If you don't have a joystick or touch tablet, you can use the keyboard cursor controls to move the pointer arrow. The keyboard isn't the fastest way to scurry across the screen, but it is exact. The joystick, however, overrides keyboard control (although you can use a properly calibrated touch tablet along with the keyboard), so you need to press the K key right away to freeze the joystick and enable keyboard control. The K key alternately freezes and reenables the joystick, and is a keyboard shortcut for the *Keyboard* command on the *Preferences* menu.

There are two ways to use the keyboard controls. The pointer can move one pixel at a time for fine movements, but it could take all day to inch your way across the

screen. If you press a cursor key rapidly or hold it down as it repeats, the cursor accelerates. It first moves one pixel at a time, then two, then three, until it's moving at the top speed of 12 pixels per keypress. If you stop pressing the cursor key, press another key, or release the key for a moment, the acceleration reverts to one pixel per keypress. If you want fine control, press and release the cursor key slowly, allowing time between each keystroke to prevent acceleration.

Keyboard Shortcuts

| Key | Menu |
|-----|------------------------|
| U | Picture/Undo |
| ^N | Picture/New |
| O | Picture/Open |
| S | Picture/Save |
| ^Q | Picture/Quit |
| D | Tools/Draw |
| L | Tools/Line |
| R | Tools/Rectangle |
| C | Tools/Circle |
| A | Tools/Airbrush |
| P | Tools/Paint |
| B | Preferences/Bkgd color |
| K | Preferences/Keyboard |
| J | Preferences/Calibrate |

Unfortunately, the program is not fast enough to keep up with the full repeat rate of the cursor keys, so even after you release the key, the program is acting on up to 15 pending keystrokes. It's best to control the number of keystrokes yourself by just pressing the same key rapidly rather than holding it down to repeat. If you do hold down the key, release it before the cursor reaches its destination. It will keep going for a short distance, then stop. With practice, you can time things right so that the cursor ends up exactly where you anticipated.

Using The Menus

To access a menu, just move the arrow cursor so that it points at one of the menu titles: *Picture*, *Tools*, or *Preferences*. The tip of the arrow is the active point, so make sure it is within the menu bar and touching the desired menu title. Now press and release the button. (The keyboard equivalent for the button is the INS key, conveniently located beneath your thumb when you use the cursor keypad.) The menu title reverses color, and the menu drops down.

Note that this differs from the

way menus are selected on other machines. On the Atari ST, menus drop down automatically if you merely point at a menu title. On the Macintosh and Amiga, you point at the menu and click to pull it down. You have to continue to hold down the button to keep the menu displayed, and move the pointer within the menu to select an item. To actually make the choice on a Mac or Amiga, you release the mouse button.

In contrast, with Screen Machine II you press and release the button to drop down the menu, move the pointer to the item you want, then press the button again to select the item and remove the menu. This technique is most appropriate when you're using pointing devices such as a joystick or cursor keys, because it's difficult to hold down a button while moving the pointer.

For example, if you point to *Picture*, then click the button, it drops down a list containing the choices *Undo*, *New*, *Open*, *Save*, *View*, and *Quit* (see figure 3). To the right of each selection are the keyboard equivalents: U, ^N, O, S, V, and ^Q. Instead of pulling down a menu and selecting a choice, you can just press the appropriate keyboard shortcut. The ^ character indicates that you should press the CTRL key along with the following character: ^N means that you should press N while holding down CTRL.

Using the more cumbersome CTRL-N and CTRL-Q sequences, instead of merely N or Q, helps prevent you from casually erasing the screen or exiting the program. Since these are destructive options, the CTRL key is used to guard against accidental keypresses. Use the accompanying table as a quick reference to keyboard shortcuts.

Selecting A Menu Item

To select a menu item, point the cursor at the desired item. As you slide the cursor up or down within a menu, the item you point to is highlighted in reverse video. You then press and release the button to select the highlighted item. To cancel a menu selection, either move the pointer outside of the menu, or move it to point at the menu title so

that no other items are selected when you press the button. If you move the cursor to the left or right of the menu border, the menu is automatically canceled. You'll hear an "uh-oh" sound effect to confirm that you've canceled the menu.

When you select an item, on the other hand, it flashes twice, emitting little tweeting sounds to let you know that you've chosen a valid option. (By the way, if you don't want any sound effects, change line 320 to read `SNDFX=0`.) After you select a menu item, some action is usually performed. If you select *New* from the *Picture* menu, for instance, the screen clears. Use *Quit* to exit the program. Following is a quick tour of the menus—we'll discuss the meaning of each item later on.

Some menu items select a setting for the program. The *Tools* menu contains the choices *Draw*, *Line*, *Rectangle*, *Circle*, *Airbrush*, and *Paint*. This menu is used to select the current drawing tool. Only one tool is active at any time. When you click the button while pointing at the drawing surface, rather than at the menu bar or within the palette, the current tool is activated, and you start the drawing action. In Draw mode, you can draw connected lines as you move the pointer about. In Line mode, though, you stretch a "rubber-band" line across the screen, emanating from the point you first clicked on. When you press the button a second time, the rubber-band line disappears, and the desired line is stamped down. In Paint mode, each click initiates a flood-fill, used to color enclosed figures.

The *Tools* menu indicates the current tool by placing a check mark next to it. A check mark can show which of several items is currently selected. If you select another drawing tool, the check mark moves to the new tool. In the *Tools* menu, a selection mutually excludes all other selections.

On the other hand, a check mark can also be used to show the status of several on/off settings. The *Preferences* menu lets you select 320×200 , 640×200 , and 320×200 PCjr drawing modes; two color palettes for the 320×200 graphics mode: *cyn/mag/wht* and

red/grn/yel; as well as *Bkgd color* (background color), *Keyboard* mode, and *Calibrate*. More than one item can be checked in the *Preferences* menu. You obviously can't draw both in both 320 × 200 mode and 640 × 200 mode at the same time, so only one of the three graphics modes is checked. However, while in 320 × 200 mode, you can choose either of the two color palettes, so both 320 × 200 would be checked as well as either *cyn/mag/wht* or *red/grn/yel*.

Ghosted Items

If you select 640 × 200 mode, some menu items under *Preferences* are no longer appropriate. It isn't possible to switch color palettes or change the background color while in 640 × 200 mode, so the inappropriate menu selections need to be disabled. Also, the 320 × 200 PCjr mode, which permits 16 colors, only works with the PCjr, so this selection should be made inaccessible when running on a PC.

Figure 1: Ghosted Items



In 640 × 200 mode, the menu items *cyn/mag/wht*, *red/grn/yel*, and *Bkgd color* are disabled, and the text of the menu items is distorted to show that you can't access them (see Figure 1). This distortion dims and garbles the text—such a menu item is *ghosted out*, as if the text was a "ghost" of the original text. The 320 × 200 PCjr option is ghosted out when running on the PC. A ghosted menu item can't be selected; you can't even highlight it by pointing to it.

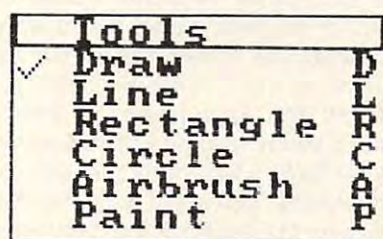
A ghosted item immediately

tells you that the menu item is inappropriate for the current environment. If you wonder which commands work in which modes, ghosting makes it obvious. Along with ghosting and the check mark, a menu can be documentation, on-line help, and a status report, as well as the device used to change these settings or activate commands.

Drawing Tools

The pointer arrow is the pass key to all the functions of Screen Machine II. You use it to select a menu item, sketch on the drawing surface, or choose a new drawing color. You already know how to use the menus. To change drawing colors, just point within the box containing the desired color and click the button. A border encloses the current drawing color so you can tell at a glance which color is being used. In 640 × 200 mode, you can only switch between black and white, of course. On the PCjr, you can select the 16-color drawing mode (see below).

Figure 2: The Tools Menu



Your default drawing tool is *Draw*, as you can confirm by looking within the *Tools* menu. With all the tools, you start the drawing action by clicking the button while pointing at the drawing area. The drawing area is bordered by the menu bar, the color palette, and is enclosed within a rectangle. You can't draw anywhere outside of this border.

In *Draw* mode, the first button click initiates the mode. The cursor disappears (to speed up drawing), and a point is plotted at the cursor position. You can now move around on the screen, leaving a trail behind, as you sketch freehand with the joystick, tablet, or cursor keys. The cursor keys are especially useful for touch-up work or small, complex figures. Again, you can use the *Keyboard* option from *Prefer-*

ences if you need to disable the joystick. If you press the button (or merely lift the stylus from the tablet surface), you exit drawing mode, and you can once again freely move the cursor without drawing on your screen canvas.

In *Line*, *Rectangle*, and *Circle* modes, the first click sets the first coordinate for the figure. You then move the pointer about to change the size or position of the figure, then press the button again to finalize the figure. While previewing the figure, the line, circle, or rectangle is repeatedly drawn and erased to allow movement. As you move the figure across the drawing surface, it may erase parts of the picture as it passes over the screen. Don't be concerned—this is just a side effect of the animation process. When you press the button to choose the desired figure, the previous screen is redrawn, restoring any erased parts, and then the desired figure is overlaid on top of the picture. If you make a mistake, you can use the *Undo* option from the *Picture* menu (or simply press U), to restore the previous screen.

In *Line* mode, the first click sets one endpoint of the line. You move the other endpoint around with your pointing device. You see how the line will look as you move it around the screen. In *Rectangle* mode, the first click sets one corner of the rectangle. As you move the pointer, you are dragging around the diagonally opposite corner of the rectangle. In *Circle* mode, the first click sets the center of the circle. You move the pointer around to enlarge or contract the radius of the circle. The second click stamps down the figure. (Remember, you can always *Undo* the most recent drawing action.)

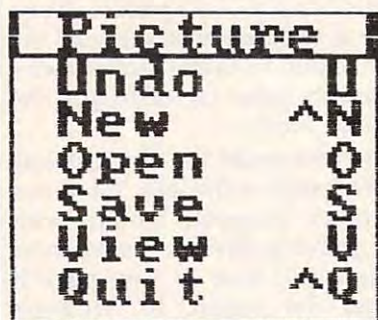
The *Airbrush* tool is handy for shading or blending colors. It randomly sprays out pixels within an 8 × 8 zone centered around the pointing arrow. The longer you stay in one place, the more paint is sprayed down. If you move around while spraying, you get a series of random dots. This approximates the behavior of a real airbrush. Again, the first click starts airbrush mode, and a second click exits airbrush mode, restoring floating cursor movement. As with *Draw*

mode, lifting the stylus from the tablet surface implies you want to stop the airbrush and go back to moving the cursor.

Use the *Paint* tool carefully. It's used to fill in an enclosed area of the screen. For example, you could draw a rectangle first, then fill it in. The paint floods out of the cursor position, and doesn't stop until it touches areas of the same color. You can only fill an area bounded by the same color as the current drawing color. If you attempt to fill with a different color, the paint overflows the container, possibly filling the whole screen. However, if you remember to press U before you start another drawing action, no harm is done.

You can use the keyboard shortcuts D for *Draw*, L for *Line*, R for *Rectangle*, C for *Circle*, A for *Airbrush*, and P for *Paint*.

Figure 3: The *Picture* Menu



The *Picture* Menu

The *Picture* menu affects the overall screen canvas. Use *Undo* to restore the previous screen. The screen is saved in a buffer before any drawing command changes the screen. *Undo* copies this buffer back onto the screen, restoring the previous screen and erasing the most recent change. Of course, *Undo* can only undo the most recent action, and you can't go back to the way the screen was before you performed the *Undo*—you can't undo an *Undo*.

The *New* option simply erases the screen. Be careful—it doesn't ask "Are you sure?" first.

Use *Quit* to exit the program. You could, of course, simply turn off the machine, but *Quit* is somewhat more elegant. Once you're back in BASIC, you can type SYSTEM to exit to DOS.

The *Save* command stores your picture on disk. *Open* restores a pre-

viously saved screen. After you select *Open* or *Save*, a box pops up in the center of the screen, prompting you to enter a filename. You can enter any legal PC-DOS filename, including a path prefix, such as A: or B:. This is the name that your picture is stored under. After you *Save* a picture, you can use *Open* to read this picture back onto the screen.

If you don't use an extender, as in FLOWERS.ART, an extender is added for you. The extender is made of the characters PI (for picture) and the number of the graphics mode used: 1 for SCREEN 1, 320 × 200; 2 for SCREEN 2, 640 × 200, and 5 for the PCjr 16-color 320 × 200 mode, SCREEN 5.

So a picture saved while in 640 × 200 mode would have the characters .PI2 appended to the filename. This extender is added for both *Open* and *Save*. If you attempt to *Open* the picture FLOWERS while in 320 × 200 mode, it actually searches for FLOWERS.PI1. However, if you're in 640 × 200 mode, it searches for FLOWERS.PI2. This prevents you from loading a picture saved in one graphics mode onto the screen of another graphics mode. If you want to defeat this, either always use an extension, as in FLOWERS.ART, or append the appropriate .PI extension. If you're loading a picture saved as FLOWERS.PI1 onto the 640 × 200 screen, you need to enter the filename FLOWERS.PI1 to prevent it from searching for FLOWERS.PI2.

If a disk error occurs, another box pops up showing you the DOS error code, and it prompts you to press either R for *Retry* or C for *Cancel*. If the error is something you can immediately correct, such as inserting a disk, you can press R to retry the disk operation. Otherwise press C to cancel the operation, then figure out what you did wrong before again selecting *Open* or *Save*.

Next month, we'll show you how to use the BLOAD command to load one of these pictures from within your own programs. If you can't wait, examine the *Open* and *Save* routines at lines 2100 and 2170.

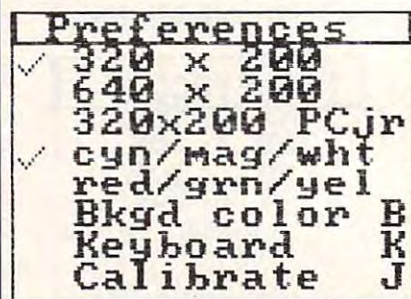
The keyboard shortcuts for the *Picture* menu are ^N for *New*, O for *Open*, S for *Save*, and ^Q for *Quit* (remember that ^ means to press CTRL as you press the indicated key).

Choosing Your Preferences

The *Preferences* menu lets you select various special options. The first three entries: 320 × 200, 640 × 200, and 320 × 200 PCjr, let you pick which graphics mode to use. (You should choose your mode before you begin drawing; the drawing area is erased when you change modes, so any drawing will be lost.) The 320 × 200 mode gives you four colors to work with. In this mode, you can choose either of two color palettes: *cyn/mag/wht* (cyan/magenta/white), or *red/grn/yel* (red/green/yellow). The latter options are ghosted in all other modes.

The 640 × 200 mode gives you more horizontal density for fine detail, but you can only choose between black and white. On the PCjr only, you can select the 320 × 200 PCjr mode and get 16 colors for vivid, realistic (or surrealistic) paintings.

Figure 4: The *Preferences* Menu



The *Bkgd color* option switches to a different background color. Each time you select it, the background color changes to the next in a series, 16 colors in all.

Some of the *Preferences* items have keyboard shortcuts: B for *Bkgd color*, K for *Keyboard*, and J for *Calibrate*.

BASIC Shortcomings

This program was an experiment of sorts, an attempt to discover if techniques such as pull-down menus can be achieved in BASIC. I knew from the beginning that BASIC's relatively slow speed (as compared to machine language or compiled languages) would be the limiting factor. In particular, using a 30,000-byte array for an *Undo* buffer causes a short delay the first time any routine is activated. The first time you try to move the cursor,

select a drawing tool, change colors, select a menu, etc., there is short delay as the huge array is shifted downward in memory to make room for new variables as they are encountered. This problem could be eliminated by referencing every variable in the program before the array is dimensioned, but this is really more trouble than it's worth.

The innermost, core routine in this program is the subroutine at line 20000, used to check for the "mouse" position. It adapts to the keyboard, joystick, and touch tablet, checks for keyboard shortcuts, scales the values to the current screen resolution, and keeps these coordinates in bounds. All this checking in such a commonly called routine is bound to slow things down. If you are using only one pointing device or one graphics mode, you may want to consider streamlining this routine to speed things up.

Screen Machine II

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!.

```
HA 120 DEFINT A-Z
PA 140 PCJR=0:ON ERROR GOTO 150:
      SOUND OFF:CLER ,,,32768!
      DEFINT A-Z:PCJR=-1
JF 150 IF NOT PCJR THEN RESUME 1
      60
OJ 160 ON ERROR GOTO 0
CN 220 DIM ARROW%(32),ZZTEMP%(64
      8)
KF 240 XMAX=250:YMAX=230:XOFF=7:
      YOFF=7
HG 250 HIGHLIGHT=2
ND 260 TRUE=-1:CURSOR=TRUE
MP 270 KEY OFF:SCREEN 0,0,0:WIDT
      H 40:COLOR ,1,1:CLS:LOCAT
      E 4,11,0:COLOR 12:PRINT "
      SCREEN MACHINE II"
MP 280 LOCATE 7,12:COLOR 10:PRIN
      T"Charles Brannon"
KD 290 COLOR 14:LOCATE 13,10:PRI
      NT"One moment, please..."
AC 300 GOSUB 9000
OB 310 SMODE=1:COLR=1:GOSUB 3000
BL 320 SNDFX=TRUE
AG 330 ACC=1:DACC=1
KG 340 FROZEN=0
BH 380 COLR=1:TOOL=1
EO 390 STRIG ON
PC 400 MX=XRES/2:MY=YRES/2:NX=MX:
      NY=MY:GOSUB 18000
LI 410 DIM UNDO%(15000)
JI 450 WHILE TRUE
BN 460 GOSUB 18000:MB=0:MNI=0
LF 470 WHILE MNI=0 AND MB=0
KF 480 GOSUB 14000
BH 490 WEND
FN 500 IF MB<>0 THEN GOSUB 10000
FH 510 IF MNI THEN GOSUB 2000
EM 520 WEND
```

```
LM 1000 WHILE MB:GOSUB 20000:WEN
      D
KB 1010 GOSUB 19000
AC 1020 IF MY>CY THEN COLR=INT(
      MX/XR#):GOSUB 6000:RETUR
      N
IC 1030 GET (1,8)-(XRES-2,CY-1),
      UNDO%
DJ 1035 SCMS=CM$:CM$=""
LG 1040 ON TOOL GOSUB 1070,1170,
      1300,1430,1560,1630
OC 1045 CM$=SCM$
JK 1050 RETURN
EA 1070 IF PENUP AND NOT KEYMODE
      THEN RETURN
KB 1080 CURSOR=0
BA 1090 WHILE MB=0 AND (NOT PENU
      P OR KEYMODE)
EB 1100 SX=MX:SY=MY:GOSUB 20000:
      MY=-MY*(MY>7 AND MY<CY)-
      8*(MY<8)-(CY-1)*(MY>=CY)
ML 1110 LINE (SX,SY)-(MX,MY),COL
      R
FK 1120 WEND
LH 1130 WHILE MB:GOSUB 20000:WEN
      D
HC 1140 CURSOR=TRUE
JM 1150 RETURN
DF 1170 SX=MX:SY=MY:CURSOR=0
MP 1180 WHILE MB=0
OD 1190 LINE (SX,SY)-(MX,MY),0
OG 1200 GOSUB 20000:MY=-MY*(MY>
      7 AND MY<CY)-8*(MY<8)-(C
      Y-1)*(MY>=CY)
BK 1210 LINE (SX,SY)-(MX,MY),CO
      LR
EA 1220 EX=MX:EY=MY
FP 1230 WEND
MM 1240 WHILE MB:GOSUB 20000:WEN
      D
NA 1250 PUT (1,8),UNDO%,PSET
HE 1260 LINE (SX,SY)-(EX,EY),COL
      R
IN 1270 CURSOR=TRUE
JH 1280 RETURN
CE 1300 SX=MX:SY=MY:CURSOR=0
LD 1310 WHILE MB=0
NK 1320 LINE (SX,SY)-(MX,MY),0,
      B
PB 1330 GOSUB 20000:MY=-MY*(MY>
      7 AND MY<CY)-8*(MY<8)-(C
      Y-1)*(MY>=CY)
FH 1340 LINE (SX,SY)-(MX,MY),CO
      LR,B
FL 1350 EX=MX:EY=MY
BK 1360 WEND
MH 1370 WHILE MB:GOSUB 20000:WEN
      D
OL 1380 PUT (1,8),UNDO%,PSET
FD 1390 LINE (SX,SY)-(EX,EY),COL
      R,B
HM 1400 CURSOR=TRUE
IS 1410 RETURN
DP 1430 SX=MX:SY=MY:CURSOR=0
MJ 1440 WHILE MB=0
PM 1450 CIRCLE (SX,SY),SQR(ABS(
      SX-MX)^2+ABS(SY-MY)^2),0
QM 1460 GOSUB 20000:MY=-MY*(MY>
      7 AND MY<CY)-8*(MY<8)-(C
      Y-1)*(MY>=CY)
NP 1470 CIRCLE (SX,SY),SQR(ABS(
      SX-MX)^2+ABS(SY-MY)^2),C
      OLR
FB 1480 EX=MX:EY=MY
GF 1490 WEND
LG 1500 WHILE MB:GOSUB 20000:WEN
      D
KE 1510 GOSUB 3000:PUT (1,8),UND
      O%,PSET
JA 1520 CIRCLE (SX,SY),SQR(ABS(
      X-EX)^2+ABS(SY-EY)^2),CO
      LR
MN 1530 CURSOR=TRUE:GOSUB 12000:
```

```
GOSUB 6000
JB 1540 RETURN
GB 1560 WHILE MB=0 AND (NOT PENU
      P OR KEYMODE)
JE 1570 GOSUB 20000:IF MY<12 OR
      MY>CY-5 THEN 1590
BG 1580 GOSUB 19000:PSET (MX+4-
      8*MRND,MY+4-8*MRND),COLR
GH 1590 WEND
OI 1600 WHILE MB:GOSUB 20000:WEN
      D
JK 1610 RETURN
NA 1630 ON ERROR GOTO 1660:PAINT
      (MX,MY),COLR:LINE (0,0)
      -(XRES-1,YRES-1),,B:GOSU
      B 6000:GOSUB 12000
GC 1640 ON ERROR GOTO 0:WHILE MB
      :GOSUB 20000:WEND
JG 1650 RETURN
KA 1660 RESUME NEXT
ED 2000 ON MNID GOSUB 2030,2320,
      2380
IP 2010 RETURN
GL 2030 ON MNID GOSUB 2060,2080,
      2100,2170,2240,2300
JI 2040 RETURN
AF 2060 GOSUB 19000:PUT (1,8),UN
      DO%,PSET:RETURN
HE 2080 GOSUB 3000:RETURN
IJ 2100 TYP$="OPEN":GOSUB 4000
OJ 2110 IF FILENAME$="" THEN 213
      0
NC 2120 ON ERROR GOTO 5500:DEF S
      EG=SEGADR:BLOAD FILENAME
      $,0
JD 2130 ON ERROR GOTO 0:CLOSE#1
BD 2140 LINE (0,0)-(XRES-1,YRES-
      1),,B:GOSUB 12000:GOSUB
      6000
JN 2150 RETURN
GC 2170 TYP$="SAVE":GOSUB 4000
NA 2180 IF FILENAME$="" THEN 221
      0
FN 2190 GET (1,8)-(XRES-2,CY-1),
      UNDO%:CLS:PUT (1,8),UNDO
      %,PSET
OL 2200 ON ERROR GOTO 5500:DEF S
      EG=SEGADR:BSAVE FILENAME
      $,0,SCREEN!
KP 2210 ON ERROR GOTO 0:CLOSE#1:
      GOSUB 3000:PUT (1,8),UND
      O%,PSET
IS 2220 RETURN
DF 2240 GOSUB 19000:CURSOR=0
ED 2250 GET (1,8)-(XRES-2,CY-1),
      UNDO%:CLS:PUT (1,8),UNDO
      %,PSET
HB 2260 WHILE MB=0:GOSUB 20000:W
      END
OB 2270 WHILE MB:GOSUB 20000:WEN
      D
MF 2280 GOSUB 3000:PUT (1,8),UND
      O%,PSET:CURSOR=-1:RETURN
CJ 2300 SCREEN 0,0,0,0:END
LM 2320 MFLAGS(MNID,TOOL)=1
DH 2330 MFLAGS(MNID,MNI)=2:TOOL
      =MNI
JO 2340 RETURN
GN 2350 STOP
NF 2380 IF MNI<4 THEN SMODE=MNI
      T-2*(MNI=3):GOSUB 3000
PK 2390 IF MNI=4 THEN COLOR ,1:
      MFLAGS(MNID,4)=2:MFLAGS(
      MNID,5)=1
BM 2400 IF MNI=5 THEN COLOR ,2:
      MFLAGS(MNID,4)=1:MFLAGS(
      MNID,5)=2
IF 2410 IF MNI=6 THEN BG=(BG+1)
      AND 15:IF SMODE=1 THEN
      COLOR BG ELSE COLOR ,BG
GF 2420 IF MNI=7 THEN FROZEN=NO
      T:FROZEN:MFLAGS(MNID,MNI
      T)=1-FROZEN
```



```

DE 2430 IF MNIT<>8 THEN RETURN
GA 2440 GOSUB 19000:LOCATE 1,1:M
SG$=LEFT$( "Move stick to
upper left, press butto
n." +SPACE$(80),SWIDTH):G
OSUB 13000
IJ 2450 WHILE STRIG(1)=0:XOFF=ST
ICK(0):YOFF=STICK(1):WEN
D
MN 2460 WHILE STRIG(1)<>0:WEND
FO 2470 LOCATE 1,1:MSG$=LEFT$( "M
ove stick to lower right
, press button." +SPACE$(
80),SWIDTH):GOSUB 13000
HK 2480 WHILE STRIG(1)=0:XMAX=ST
ICK(0):YMAX=STICK(1):WEN
D
MG 2490 WHILE STRIG(1)<>0:WEND
DH 2500 XRATIO#=(XRES/XMAX):YRATIO
#=(YRES/YMAX)
IF 2510 GOSUB 12000:RETURN
KA 3000 GOSUB 19000
KO 3010 IF SMODE=PMODE THEN 3030
KO 3020 ON SMODE GOSUB 3110,3150
,3030,3030,3190
CP 3030 PMODE=SMODE
AG 3040 SWIDTH=INT(XRES/8):XRATI
O#=(XRES/XMAX):YRATIO#=(Y
RES/YMAX)
CG 3050 CLS:PSET (10,10):DRAW "b
m10,10d3e313f5":GET (10,
10)-(17,17),ARROW%
HI 3060 XARROW=8:YARROW=8
QL 3070 CLS:LINE (0,0)-(XRES-1,Y
RES-1),,B
PC 3080 GOSUB 6000:GOSUB 12000
XI 3090 RETURN
DD 3110 SCREEN 1:COLOR 0,1:COLR=
1:XRES=320:YRES=200:BG=0
:MAXCOLOR=4
HA 3120 GOSUB 3230:MFLAGS(3,1)=2
:SEGADR=&H8800:SCRLEN!=1
6384
PJ 3130 MFLAGS(3,4)=2:MFLAGS(3,5
)=1:MFLAGS(3,6)=1
JL 3140 RETURN
QN 3150 SCREEN 2:XRES=640:YRES=2
00:MAXCOLOR=2:COLR=1
JG 3160 GOSUB 3230:MFLAGS(3,2)=2
:SEGADR=&H8800:SCRLEN!=1
6384
ID 3170 MFLAGS(3,4)=0:MFLAGS(3,5
)=0:MFLAGS(3,6)=0
JH 3180 RETURN
JK 3190 SCREEN 5:XRES=320:YRES=2
00:MAXCOLOR=16:COLR=1
FE 3200 GOSUB 3230:MFLAGS(3,3)=2
:SEGADR=&H1800:SCRLEN!=3
2768!
KB 3210 MFLAGS(3,4)=0:MFLAGS(3,5
)=0:MFLAGS(3,6)=1
IH 3220 RETURN
JC 3230 MFLAGS(3,1)=1:MFLAGS(3,2
)=1:MFLAGS(3,3)=PCJR:RE
TURN
DL 4000 GOSUB 19000:GET (1,8)-(X
RES-2,CY-1),UNDO%
JO 4010 MSG1$="Please enter name
":MSG2$="of picture to "
+TYP$
FF 4020 TW=SWIDTH/2-10:LINE (TW*
8-10,50)-(TW*8+160,100),
0,BF:LINE (TW*8-10,50)-(
TW*8+160,100),,B:LINE (T
W*8-8,52)-(TW*8+158,98),
,B
HD 4030 LOCATE 8,SWIDTH/2-LEN(MS
G1$)/2:PRINT MSG1$:LOCAT
E 9,SWIDTH/2-LEN(MSG2$)/
2:PRINT MSG2$
PG 4040 LINE (TW*8-5,78)-(TW*8+1
55,89),,B:LOCATE 11,TW+1
:MAXLEN=18:GOSUB 5000
IP 4050 FILENAME$=EDT$:IF FILENA
ME$>" " THEN IF MID$(EDT$
,LEN(EDT$)+3*(LEN(EDT$)>
3),1)<>"." THEN FILENAME
$=FILENAME$+"."PI"+CHR$(4
8+SMODE)
NC 4060 PUT (1,8),UNDO%,PSET
JD 4070 RETURN
QP 5000 EDT$="":IX=POS(0):IY=CSR
LIN:XI=IX:KBD=-1:IF MAXL
EN=0 THEN MAXLEN=79-IX
GB 5010 WHILE KBD<>13
GE 5020 XI=LEN(EDT$)+IX:LOCATE
IY,XI:PRINT "_";:KBD$=IN
PUT$(1)
PF 5030 KBD=ASC(KBD$):LOCATE IY
,XI:PRINT " ";
DC 5040 IF KBD=8 AND LEN(EDT$)>
0 THEN EDT$=LEFT$(EDT$,L
EN(EDT$)-1)
KH 5050 IF LEN(EDT$)<MAXLEN AND
(KBD AND 127)>=32 THEN
EDT$=EDT$+KBD$:LOCATE IY
,XI:PRINT KBD$;
GI 5060 WEND
JE 5070 RETURN
JL 5500 CLOSE #1
EJ 5510 GOSUB 19000:GET (1,8)-(X
RES-2,CY-1),UNDO%
GA 5520 TW=SWIDTH/2-10:LINE (TW*
8-10,50)-(TW*8+160,100),
0,BF:LINE (TW*8-10,50)-(
TW*8+160,100),,B:LINE (T
W*8-8,52)-(TW*8+158,98),
,B
PK 5530 IF ERR=52 THEN MSG1$="D
OS ERROR #"+STR$(ERR):EL
SE MSG1$="ERROR #"+STR$(
ERR)+" in line"+STR$(ERL
)
JF 5540 MSG2$="(R)etry or (C)anc
el"
PD 5550 LOCATE 8,SWIDTH/2-LEN(MS
G1$)/2:PRINT MSG1$:LOCAT
E 10,SWIDTH/2-LEN(MSG2$)
/2:PRINT MSG2$
BF 5560 KBD$=INPUT$(1):IF KBD$<
"r" AND KBD$<"R" AND KB
D$<"c" AND KBD$<"C" TH
EN 5560
QA 5570 PUT (1,8),UNDO%,PSET
HL 5580 IF KBD$="r" OR KBD$="R"
THEN RESUME ELSE RESUME
NEXT
FH 6000 XR#=(XRES/MAXCOLOR):CH=11:
CY=YRES-CH-1
PH 6010 LINE (0,CY)-(XRES-1,YRES
-1),0,BF
EJ 6020 FOR I=0 TO MAXCOLOR-1
DA 6030 LINE (I*XR#+2,CY+3)-(I*X
R#+XR#+3,CY+CH-3),I,BF
QN 6040 NEXT
BH 6050 LINE (0,CY)-(XRES-1,YRES
-1),,B
JD 6060 LINE (COLR*XR#,CY+2)-(CO
LR*XR#+XR#+1,CY+CH-2),,B
JF 6070 RETURN
BK 9000 RESTORE 9090
GO 9010 WHILE MNSTR$<>"x"
NJ 9020 READ MNID,MNIT,MFLAG,MN
STR$
ME 9030 IF MNSTR$<>"x" THEN GOS
UB 11000
FG 9040 WEND
CN 9050 MFLAGS(3,3)=PCJR
BM 9060 CM$="U11"+CHR$(14)+"1201
3514V15"+CHR$(17)+"16D21
L22R23C24A25P26B36K37J38
"
KI 9070 RETURN
BD 9090 DATA 1,0,1,"Picture "
HE 9100 DATA 1,1,1,"Undo U"
IN 9110 DATA 1,2,1,"New ^N"
LH 9120 DATA 1,3,1,"Open O"
PK 9130 DATA 1,4,1,"Save S"
EM 9140 DATA 1,5,1,"View V"
EK 9150 DATA 1,6,1,"Quit ^Q"
ND 9170 DATA 2,0,1,"Tools
"
HJ 9180 DATA 2,1,2," Draw D
"
ML 9190 DATA 2,2,1," Line L
"
PH 9200 DATA 2,3,1," Rectangle R
"
ID 9210 DATA 2,4,1," Circle C
"
JC 9220 DATA 2,5,1," Airbrush A
"
PI 9230 DATA 2,6,1," Paint P
"
GP 9250 DATA 3,0,1,"Preferences
"
HG 9260 DATA 3,1,2," 320 x 200
"
OI 9270 DATA 3,2,1," 640 x 200
"
KO 9280 DATA 3,3,0," 320x200 PCj
r"
DB 9290 DATA 3,4,2," cyn/mag/wht
"
JI 9300 DATA 3,5,1," red/grn/yel
"
JN 9310 DATA 3,6,1," Bkgd color
B"
ED 9320 DATA 3,7,1," Keyboard
K"
IB 9330 DATA 3,8,1," Calibrate
J"
OD 9340 DATA ,,,x
LJ 11000 MAXMENUS=8:MAXITEMS=8
GF 11010 IF NOT MENUINIT THEN DI
M MTITLE$(MAXMENUS,MAXI
TEMS),MFLAGS(MAXMENUS,M
AXITEMS),MITEMS(MAXMENU
S),MSAVE$(800*MAXITEMS+
8),MX(MAXMENUS):TOPID=0
:MENUINIT=-1
JE 11020 IF MNID<1 OR MNID>MAXME
NUS OR MNIT<0 OR MNIT>M
AXITEMS THEN PRINT "ILL
EGAL MENU PARAMETERS":S
TOP
IH 11030 MTITLE$(MNID,MNIT)=MNST
R$:MFLAGS(MNID,MNIT)=MF
LAG
OL 11040 IF MNIT>MITEMS(MNID) TH
EN MITEMS(MNID)=MNIT
BA 11050 IF MNID>TOPID THEN TOPI
D=MNID
IF 11060 RETURN
QJ 12000 IF SWIDTH=0 THEN IF XSI
ZE THEN SWIDTH=INT(XSIZ
E/8+.5) ELSE SWIDTH=80
DM 12010 MSG$=" ":MX(0)=8:SVX=PO
S(0):SVY=CSRLIN
GL 12020 FOR MI=1 TO TOPID:MX(MI
)=MX(MI-1)+8+LEN(MTITLE
$(MI,0)):8:MSG$=MSG$+"
"+MTITLE$(MI,0):NEXT:MS
G$=MSG$+SPACE$(SWIDTH-L
EN(MSG$))
PF 12030 LOCATE 1,1:GOSUB 13000
HC 12040 LOCATE SVY,SVX:RETURN
BI 13000 X1=POS(0):8-8:Y1=CSRLIN
:8-8:PRINT MSG$:X2=X1+
LEN(MSG$):8-1:IF X2>=SW
IDTH:8 THEN X2=SWIDTH:8
-1
LL 13010 GET (X1,Y1)-(X2,Y1+7),Z
ZTEMP$:PUT (X1,Y1),ZTE
MP$,PRESET:RETURN
ND 14000 XSAVE=POS(0):YSAVE=CSRL
IN
DC 14010 MNIT=0:MNID=0:GOSUB 200
00

```



```

LM 14020 IF MY>7 OR MB=0 THEN RE
TURN
FM 14030 WHILE MB:GOSUB 20000:WE
ND
EH 14040 MI=1:WHILE MI<=TOPID AN
D NOT (MX>=MX(MI-1) AND
MX<=MX(MI)):MI=MI+1:WE
ND
DG 14050 IF MI>TOPID THEN RETURN
CF 14060 MNID=MI
HJ 14070 IF SNDFX THEN SOUND 100
00,.5
QI 14080 GOSUB 16000:GOSUB 20000
FE 14090 SAVDACC=DACC:SAV$=CM$:C
M$="":IF KEYMODE THEN M
Y=2:NY=MY:DACC=-B
DN 14100 WHILE MX>=MX(MNID-1) AN
D MX<=MX(MNID) AND MB=0
GOSUB 20000
KI 14110 MI=INT(MY/8):IF MI>MIT
EMS(MNID) THEN GOTO 141
50
HD 14120 IF MI=MNIT OR MFLAGS(M
NID,MI)=0 THEN 14180
AF 14130 GOSUB 19000
CN 14140 IF MNIT>0 THEN LOCATE
MNIT+1,INT(MX(MNID-1)/8
+2):PRINT MTITLE$(MNID,
MNIT)
LG 14160 IF MI>0 AND MI<=MITEMS
(MNID) THEN MNIT=MI:LOC
ATE MNIT+1,INT(MX(MNID-
1)/8)+2:MSG$=MTITLE$(MN
ID,MNIT):GOSUB 13000:IF
SNDFX THEN SOUND 20000
,.1
OD 14170 IF MI>MITEMS(MNID) THEN
MNIT=0
LL 14180 WEND
BE 14190 IF MX<MX(MNID-1) OR MX>
MX(MNID) THEN MNIT=0
HE 14200 IF MNIT THEN GOSUB 1500
0
GO 14210 GOSUB 17000
FO 14220 WHILE MB:GOSUB 20000:WE
ND
NG 14240 IF MNIT=0 THEN MNID=0:I
F SNDFX THEN SOUND 150,
2:SOUND 50,1
CK 14250 GOSUB 18000:DACC=SAVDAC
C:CM$=SAV$:LOCATE YSAVE
,XSAVE
JB 14260 RETURN
JM 15000 IF MNIT=0 OR HIGHLIGHT=
0 THEN RETURN
PC 15010 MSG$=MTITLE$(MNID,MNIT)
:FOR MI=1 TO HIGHLIGHT:
LOCATE MNIT+1,XP:GOSUB
13000
ON 15020 IF SNDFX THEN SOUND 100
00+MI*500,.1
LA 15030 LOCATE MNIT+1,XP:PRINT
MSG$
BN 15040 NEXT:RETURN
KE 16000 WX1=MX(MNID-1):WX2=MX(M
NID):WY1=8:WY2=8+B*MITE
MS(MNID):XP=INT(WX1/8)+
2
II 16010 GOSUB 19000
CL 16020 LOCATE 1,XP-1:PRINT " "
+MTITLE$(MNID,0)
EE 16030 GET (WX1-2,WY1)-(WX2+2,
WY2+2),MSAVE%
MN 16040 LINE (WX1-2,WY1-1)-(WX2
+2,WY2+2),B
IL 16050 LINE (WX1-1,WY1)-(WX2+1
,WY2+1),0,BF
MM 16060 FOR MI=1 TO MITEMS(MNID
)
JO 16070 LOCATE MI+1,XP:PRINT M
TITLE$(MNID,MI)
DL 16080 IF MFLAGS(MNID,MI)=2 T

```

```

HEN PSET (WX1,MI*8+5):D
RAW "f2e5"
CA 16090 IF MFLAGS(MNID,MI)=0 T
HEN GET (WX1,MI*8)-(WX1
+LEN(MTITLE$(MNID,MI))*
8+7,MI*8+7),ZZTEMP$:PUT
(WX1,MI*8),ZZTEMP$,PSE
T:PUT (WX1+1,MI*8),ZZTE
MP%
QL 16100 NEXT MI
IO 16110 RETURN
HS 17000 GOSUB 19000
QK 17010 PUT (WX1-2,WY1),MSAVE%,
PSET
AB 17020 LOCATE 1,XP-1:MSG$=" "
+MTITLE$(MNID,0):GOSUB 1
3000
IF 17030 RETURN
LA 18000 IF CURSOR=0 OR TOGGLE=1
THEN RETURN
HJ 18010 PUT (MX,MY),ARROW%:TOGG
LE=1:RETURN
JF 19000 IF CURSOR=0 OR TOGGLE=0
THEN RETURN
FK 19010 PUT (MX,MY),ARROW%:TOGG
LE=0:RETURN
DN 20000 MB=0:PENUP=0
JL 20010 IF NOT FROZEN THEN S0=S
TICK(0):S1=STICK(1):MB=
STRIG(1):IF S0<>XOFF OR
S1<>YOFF THEN NX=INT((
S0-XOFF)*XRATIO#):NY=IN
T((S1-YOFF)*YRATIO#):KE
YMODE=0:ELSE PENUP=-1
JN 20020 MK$=INKEY$:KY=0:IF MK$=
" " THEN IF TIMER>TM!
THEN ACC=ABS(DACC):TM!=
TIMER+.1:GOTO 20060 ELS
E 20060
LN 20025 KY=ASC(MID$(MK$,2)+CHR$
(0)):MB=MB OR -(KY=82):
KEYMODE=-1
EE 20030 NX=-(NX+ACC*(KY=75)-ACC
*(KY=77))*((KY<>71):NY=
-(NY+ACC*(KY=72)-ACC*(KY
=80))*((KY<>71)
MA 20040 IF KY=PK THEN ACC=ACC+2
*(ACC<13)*(DACC>0):PK=K
Y:ELSE ACC=ABS(DACC):PK
=KY
DC 20050 KY=ASC(MK$):IF NOT (KY>
47 AND KY<58) THEN WHER
E=INSTR(CM$,CHR$(KY+32*
(KY>96 AND KY<123))):IF
WHERE THEN MNID=VAL(MI
D$(CM$,WHERE+1,1)):MNIT
=VAL(MID$(CM$,WHERE+2,1
)):IF MFLAGS(MNID,MNIT)
=0 THEN MNIT=0:MNID=0 E
LSE GOSUB 21010
HK 20060 IF NX=MX AND NY=MY THEN
RETURN
OP 20065 XBOUND=XRES-XARROW:YBOU
ND=YRES-YARROW
EI 20070 NX=-NX*(NX>0 AND NX<=XB
OUND)-XBOUND*(NX>XBOUND
)-(NX<1)
FO 20080 NY=-NY*(NY>0 AND NY<=YB
OUND)-YBOUND*(NY>YBOUND
)-(NY<1)
PN 20090 GOSUB 19000:MX=NX:MY=NY
:GOSUB 18000
HP 20100 RETURN
JJ 21010 XP=INT(MX(MNID-1)/8)+2:
MSG$=" "+MTITLE$(MNID,0
):GOSUB 19000
GG 21015 LOCATE 1,XP-1:PRINT MSG
$
NI 21020 IF SNDFX THEN SOUND 100
00,.1
FL 21030 LOCATE 1,XP-1:GOSUB 130
00
IO 21040 RETURN

```

Attention Programmers

COMPUTE! magazine is currently looking for quality articles on Commodore, Atari, Apple, and IBM computers (including the Commodore Amiga and Atari ST). If you have an interesting home application, educational program, programming utility, or game, submit it to COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Or write for a copy of our "Writer's Guidelines."

COMPUTE! Subscriber Services

Please help us serve you better. If you need to contact us for any of the reasons listed below, write to us at:

COMPUTE! Magazine
P.O. Box 10954
Des Moines, IA 50340

or call the Toll Free number listed below.

Change Of Address. Please allow us 6-8 weeks to effect the change; send your current mailing label along with your new address.

Renewal. Should you wish to renew your COMPUTE! subscription before we remind you to, send your current mailing label with payment or charge number or call the Toll Free number listed below.

New Subscription. A one year (12 month) US subscription to COMPUTE! is \$24.00 (2 years, \$45.00; 3 years, \$65.00. For subscription rates outside the US, see staff page). Send us your name and address or call the Toll Free number listed below.

Delivery Problems. If you receive duplicate issues of COMPUTE!, if you experience late delivery or if you have problems with your subscription, please call the Toll Free number listed below.

COMPUTE!
1-800-247-5470
In IA 1-800-532-1272

Loading And Linking Commodore Programs

Part 5 The Commodore 128

Jim Butterfield, Associate Editor

This month's installment concludes the series by discussing load/link techniques on Commodore's newest eight-bit computer, the 128. As you'll see, the 128's powerful BASIC has simple, built-in commands to perform jobs that require programming tricks on earlier Commodore computers.

There are three major ways to connect programs together. *Chaining* allows several programs to perform a job, each program continuing the work that a previous program has done. *Load linking* enables one program to call another, with the new program starting fresh on a new task. *Overlaying* allows a main program to call in supplementary material, such as machine language subroutines, data tables, or additional screens. All these techniques are easy on the Commodore 128 in 128 mode. For 64 mode, of course, you can use the techniques explained in previous articles in this series.

Chaining

A program that is chained is broken into separate modules, and each part runs separately. The programs may proceed in a specific order; for

example, an input program may be followed by a sorting program and then an output program. Or a menu program may call in other programs that you request.

The 128's DLOAD command makes disk chaining extraordinarily easy. If a program executes the statement DLOAD "THISPROG", the computer loads and runs the program named THISPROG. Variables from the earlier program are preserved, so the new program can continue where the old one left off.

The chaining pitfalls of earlier Commodore machines don't apply to the 128. Because the 128 stores the BASIC program text in a different bank of RAM from the working values (variables, strings, and arrays), there is no danger that DLOAD will interfere with variables. The new program simply replaces the old one.

By the way, the 128 has no static strings; all strings, whether static or dynamic, are safely stored in bank 1.

Let's revise the rules for well-chained programs on the 128:

- It doesn't matter whether the first program is bigger or smaller than subsequent programs.

- Strings, variables, and arrays are passed from program to program.
- If you use DEF FN definitions, redefine them in each program module.
- Arrays should be DIMensioned only once, preferably in the first program.

A Short Example

Let's write the first of a small series of Commodore 128 programs which chain together. We'll call our first program START, and it assumes that you want to record grades for eight students.

```
110 PRINT "SIMPLE GRADEBOOK  
    DEMO"  
120 DIM N$(15),M(15)  
130 N=8  
140 FOR J=1 TO N  
150 PRINT "STUDENT";J;  
160 INPUT "NAME";N$(J)  
170 INPUT "SCORE";M(J)  
180 NEXT J  
190 DLOAD "MENU"
```

When the program runs to this point, we have data on eight students. Save the program using the filename START.

Now let's enter the menu program. Type NEW and enter this:

```
100 PRINT  
110 PRINT "DO YOU WANT TO--  
    "
```



```

120 PRINT "1. CALCULATE AVE
    RAGE SCORE"
130 PRINT "2. CALCULATE HIG
    H/LOW SCORES"
140 PRINT "3. QUIT"
150 PRINT
160 INPUT "YOUR CHOICE (1-3
    )";C
170 ON C GOTO 300,310,320
180 GOTO 160
300 DLOAD "C.AVG"
310 DLOAD "C.HIL"
320 END

```

Note that line 300 won't run into line 310, nor 310 into 320. The moment the program executes DLOAD, the new program loads and begins running. After checking this program closely, save it on disk with the filename MENU. (The name is important; don't substitute any other filename.)

Now type NEW and enter this program:

```

100 PRINT
110 A=0
120 FOR J=1 TO N
130 A=A+M(J)
140 NEXT J
150 PRINT "AVERAGE SCORE FO
    R";N;"STUDENTS =" ;A/N
160 PRINT
170 DLOAD "MENU"

```

Check this closely and save it as C.AVG. Again, the filename is important. Now type NEW and enter this program:

```

100 PRINT
110 H=M(1):L=M(1)
120 FOR J=1 TO N
130 IF H<M(J) THEN H=M(J)
140 IF L>M(J) THEN L=M(J)
150 NEXT J
160 PRINT "HIGH SCORE WAS";
    H;"BY:"
170 FOR J=1 TO N
180 IF H=M(J) THEN PRINT N$
    (J)
190 NEXT J
200 PRINT "LOW SCORE WAS";L
    ;"BY:"
210 FOR J=1 TO N
220 IF L=M(J) THEN PRINT N$
    (J)
230 NEXT J
240 PRINT
250 DLOAD "MENU"

```

Again, check your typing closely and save it as C.HIL to complete the set.

Now you can experiment with chaining on the 128 by loading the first program (filename START). Note that this program is smaller than both MENU and C.HIL. On earlier Commodore computers, that would be a problem. But it doesn't matter on the 128.

Load Linking

Chaining links one program to the next while keeping the first program's working values intact. That's useful when you're continuing a calculation. But sometimes you'd rather throw away these values, allowing the newly loaded program to start fresh. The RUN command does exactly that. No fuss or bother—just specify the appropriate program name and you're in business. The old variables disappear, the pointers are reset, and the new program starts running.

To illustrate, let's write two very simple programs and use a menu program to select which one to use. Type NEW, then enter this simple square root program:

```

100 PRINT "TABLE OF SQUARE
    {SPACE}ROOTS"
110 FOR J=1 TO 20
120 PRINT J, SQR(J)
130 NEXT J

```

You can try running the program if you want. Save it with the filename SQUARE.

Now type NEW again and enter this simple cube root program:

```

100 PRINT "TABLE OF CUBE RO
    OTS"
110 X=1/3
120 FOR I=1 TO 20
130 PRINT I, I^X
140 NEXT I

```

Again, you might like to try running the program. Save it with the filename CUBE. Type NEW again and enter this simple 128 loading program:

```

100 DATA SQUARE,CUBE
110 READ A$(1),A$(2)
120 PRINT "WHICH ROOTS DO Y
    OU WANT--"
130 FOR J=1 TO 2
140 PRINT J;A$(J)
150 NEXT J
160 INPUT "WHICH (1 OR 2)";
    N
170 IF N<1 OR N>2 GOTO 120
180 RUN (A$(N))

```

Note the syntax of the RUN command. If you don't specify a drive number, the computer assumes you want drive 0. If you want to run a program on a disk in drive 1, you would add ,D1 to the end of the filename. And if you want to use a variable for the filename (as shown above), it must be enclosed in parentheses.

When you run the menu program, it loads and runs SQUARE or

CUBE as selected. When the new program runs, all the old variables are scrapped. The second program starts fresh.

Overlaying

This technique brings in extra material to accompany a BASIC program. It might be a machine language routine, a screen, sprite shapes, or data tables. Whereas chaining and load linking move from one BASIC program to another, overlaying lets the same BASIC program continue with new data in memory.

On the Commodore 128, BASIC 7.0's BLOAD command can bring in the material with no problems. It loads the file, and the BASIC program continues with the next statement. It's quite straightforward, especially compared to the gyrations required on earlier Commodore machines.

However, you must take care not to BLOAD information into the same area of memory occupied by the BASIC program itself (a crash usually results). BLOAD lets you specify a load address, but it's usually convenient to BLOAD a file into the same memory area from which it was saved.

Here's a quick example. First, let's set up a short machine language routine that prints a string of characters. Type NEW and enter this program:

```

100 DATA 0,26
110 DATA 162,65
120 DATA 138
130 DATA 32,210,255
140 DATA 232
150 DATA 201,90
160 DATA 144,247
170 DATA 169,13
180 DATA 76,210,255
190 A=18
200 FOR J=1 TO A
210 READ X
220 T=T+X
230 NEXT J
240 IF T<>2525 THEN STOP
250 RESTORE
260 DOPEN#8,"ML,P",W
270 FOR J=1 TO A
280 READ X
290 PRINT#8,CHR$(X);
300 NEXT J
310 CLOSE 8

```

Be sure that line 290 ends with a semicolon. Then run the program. If it stops at line 240, there's a typing error in one of the DATA statements. Otherwise, it creates a one-block machine language routine on

disk called ML. As we'll see in a moment, this routine prints the alphabet on the screen when called into memory.

Here's our main program, which loads the ML module we just created:

```
100 BANK 15
110 BLOAD "ML"
120 PRINT "HERE IS THE ALPH
    ABET---"
130 SYS 6656
140 PRINT "HERE IT IS AGAIN
    ..."
150 SYS 6656
160 PRINT "THAT'S ALL."
```

BLOAD just brings the ML routine into memory and makes it available to the BASIC program. Simple and effective.

Overlays are popular with machine language programmers on the 128 partly because they are so easy to do and partly because of the mobility of BASIC programs. Depending on recent graphics activities, a BASIC program might start at address 7169 (the usual place) or at 16385 (if a graphics area has been allocated). Rather than puzzle over how to fit a machine language routine into memory with these uncertain locations, many programmers use an overlay. That way they know where the routine loads, even if they're not sure where the BASIC program might be.

Compared to the complexity of earlier Commodore computers, these techniques are a snap on the Commodore 128 in 128 mode. Just remember: DLOAD for chaining, RUN for load linking, and BLOAD for overlaying. ©

Copyright 1986, Jim Butterfield

Attention Programmers

COMPUTE! magazine is currently looking for quality articles on Commodore, Atari, Apple, and IBM computers (including the Commodore Amiga and Atari ST). If you have an interesting home application, educational program, programming utility, or game, submit it to COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Or write for a copy of our "Writer's Guidelines."

Apple ProDOS Catalog Sorter

William J. Coohon

Here's a short utility program that helps you organize your floppy disks by displaying or printing sorted directories. It works on any Apple II-series computer with ProDOS.

Sometimes, locating a particular program or file within a large collection of disks is like searching for a contact lens in a bowl of water—especially when any list of your disk directories will inevitably be outdated and in no particular order. "ProDOS Catalog Sorter" helps you eliminate the confusion by sorting ProDOS directories and listing them on your monitor or printer.

Type in the program below and save a copy before running it for the first time. There are two options to consider when typing Catalog Sorter: date and time stamping, and printer set-up. Stamping the date and time on listings is extremely useful for keeping track of how current (or how old) the information is. If your Apple doesn't have a clock, you may remove certain lines from Catalog Sorter or use the date/time-setting program found in the "Reader's

Feedback" column in November, 1985 COMPUTE!. Without the date and time, the program prints zeros. To remove this feature, delete lines 280-320, 345, 445, and 780.

You can also determine how your printer should generate hardcopies of the sorted directories. My directory listings are printed at 17 characters per inch with 8 lines per inch spacing. That way, they can be trimmed down to fit neatly inside a disk envelope or storage case. The printer control characters for an Apple Imagewriter are set up in line 440. If you want to substitute your own printer options, simply alter these codes. If you want your printer to use its defaults, delete line 440 completely. The variable P in line 440 is set to a value of 1 to allow the program to reset the printer options later in line 560, which may also have to be altered for other printer control characters.

Sorting Directories

When you run the program, Catalog Sorter prompts you for drive number 1 or 2 (to exit the program at this stage, simply enter 0). Next, you are asked whether you want

the directories sorted. Type N to disable sorting; any other response sorts your directories in alphabetical order. When might you want to disable sorting? Sometimes programs or files are grouped under directories logically, according to their respective functions. But other files—for instance, monthly financial data—might be organized chronologically. Sorting such files alphabetically would make the grouping less meaningful.

After reading the disk directory, the program asks if you wish to view the listing on your screen or route the output to your printer. It tells you which directory is being sorted (if you choose to sort), then prints the list and moves on to the next directory (if any others exist). Multiple directories are read, sorted, and listed separately to maintain the order of the directory hierarchy. The bottom line of the directory list indicates how many disk blocks are free, how many are used, and the total available (see figure). When all directories on a disk are read, sorted, and listed, the program gives you the option to quit or repeat the process.

If the directory list is displayed on the screen, it appears in the same format as if you typed ProDOS CATALOG command—in 80 columns (note that the abbreviated CAT command uses a 40-column format). This is rather difficult to read on a 40-column display, so an 80-column screen is recommended

unless you're interested mainly in the hardcopy listings. Line 260 sets up S\$ to switch to the 80-column screen, assuming that the 80-column hardware is addressed at slot 3 (the normal slot for the IIe and IIc). If you wish, you can modify Catalog Sorter to display only 40 columns on each line: Change the PR#3 in line 260 to PR#0.

Apple ProDOS Catalog Sorter

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" in this issue of COMPUTE!

```

28 100 DIM SK$(100),DA$(100),SD$
      (10)
13 110 GOTO 260
00 120 S = E
0F 130 S = INT (S / 2): PRINT ".
      ";
CC 140 IF S = 0 THEN 720
FC 150 K = E - S: J = 1
65 160 I = J
34 170 L = I + S
BB 180 IF SK$(I) < SK$(L) THEN
      230
6F 190 T$ = SK$(I):SK$(I) = SK$(
      L):SK$(L) = T$
F1 200 U$ = DA$(I):DA$(I) = DA$(
      L):DA$(L) = U$
91 210 I = I - S
73 220 IF I > 1 THEN 170
2C 230 J = J + 1
F0 240 IF J < K THEN 160
98 250 GOTO 130
19 260 D$ = CHR$(4):ES$ = CHR$
      (27):P$ = D$ + "PR#1":S$
      = D$ + "PR#3"
5D 270 P = 0
31 280 D = PEEK (49040) - INT (
      PEEK (49040) / 32) * 32
40 290 Y = INT ( PEEK (49041) /
      2)
7E 300 M = ( PEEK (49041) - Y *
      2) * 8 + INT ( PEEK (4904
      0) / 32)

```

```

89 310 MI = PEEK (49042):H = PEE
      K (49043)
AE 320 MI$ = STR$(MI): IF MI <
      10 THEN MI$ = "0" + MI$
51 330 E = 0:C = 0:F = 0: PRINT
      S$: PRINT
F6 340 HOME : HTAB 32: PRINT "Pr
      oDOS CATALOG SORTER"
6C 345 GOSUB 780
58 350 VTAB 8: INPUT "Drive <1>
      or <2>, <0> will END? ";N
AD 360 IF N = 0 THEN GOTO 560
92 370 IF N < 1 OR N > 2 THEN 34
      0
43 380 PRINT : INPUT "<S> sort o
      r <N> no sort? ";B$
76 390 IF B$ < "N" AND B$ < "
      n" THEN B$ = "S"
EC 400 GOSUB 600: GOSUB 620
E9 410 PRINT : INPUT "<S> screen
      or <P> printer? ";A$
E3 420 IF A$ < "P" AND A$ < "
      p" THEN HOME : GOTO 450
AA 430 A$ = "P": PRINT P$
E5 440 IF P = 0 THEN PRINT CHR$
      (9)"136N": PRINT ES$: CHR
      $(81):ES$: CHR$(66):P =
      1
6D 445 GOSUB 780
17 450 PRINT : PRINT L1$: PRINT
93 460 PRINT L2$: PRINT L3$: GOS
      UB 730
93 470 IF C = 0 THEN 510
7A 480 FOR ID = 1 TO C
4D 490 GOSUB 600:L1$ = L1$ + SD$
      (ID): GOSUB 620
0F 500 PRINT : GOSUB 580: PRINT
      L1$: GOSUB 730: NEXT
1C 510 GOSUB 580: PRINT LEFT$(L
      5$,64);" FILES: "F
C9 520 IF A$ = "P" THEN PRINT S$
5C 530 PRINT : INPUT "MORE (Y,N)
      ? ";A$
31 540 IF A$ = "N" OR A$ = "n" T
      HEN 560
9C 550 GOTO 330
DA 560 IF P = 1 THEN PRINT P$: P
      RINT ES$: CHR$(99): PRIN
      T S$
8A 570 PRINT "Bye!": END
A5 580 IF A$ = "P" THEN PRINT P$
27 590 RETURN
CB 600 PRINT D$:"PREFIX,D"N
D0 610 PRINT D$:"PREFIX": INPUT L
      1$: RETURN
37 620 PRINT D$:"OPEN "L1$,"TDIR"
F4 630 PRINT D$:"READ "L1$
F0 640 INPUT L1$:E = 0
AC 650 INPUT L2$: INPUT L3$
E4 660 INPUT L4$
D8 670 IF L4$ = "" THEN 710
46 680 E = E + 1:SK$(E) = MID$(
      L4$,2,15):DA$(E) = L4$
74 690 IF MID$(L4$,18,3) = "DIR
      " THEN C = C + 1:SD$(C) =
      MID$(L4$,2,15)
19 700 GOTO 660
63 710 IF B$ = "S" THEN PRINT "N
      ow sorting "L1$".";: GOTO
      120
4C 720 INPUT L5$: PRINT : PRINT
      D$:"CLOSE ": RETURN
9F 730 GOSUB 580:F = F + E
E8 740 IF E > 0 THEN FOR I = 1 T
      O E: PRINT DA$(I): NEXT
7E 750 IF E = 0 THEN PRINT " (D
      IRECTORY EMPTY)"
A8 760 PRINT : IF A$ = "P" THEN
      PRINT S$
25 770 RETURN
E8 780 HTAB 28: VTAB 4: PRINT "D
      ATE: "M"/"D"/"Y" TIME:
      "H": "MI$: RETURN

```

A sample directory listed generated with "ProDOS Catalog Sorter."

DATE: 12/30/85 TIME: 19:00

/COOHON,30DEC85

| NAME | TYPE | BLOCKS | MODIFIED | CREATED | ENDFILE | SUBTYPE |
|------------------|-----------------|-------------------|-----------------|-----------------|---------|---------|
| *BASIC.SYSTEM | SYS | 21 | (NO DATE) | 8-DEC-85 22:00 | 10240 | |
| CAT.SORT | BAS | 5 | 30-DEC-85 18:30 | 30-DEC-85 18:30 | 1566 | |
| CAT.SORT.COPY2 | BAS | 5 | 30-DEC-85 18:30 | 30-DEC-85 18:30 | 1566 | |
| CAT.SORT.TEXT | TXT | 8 | 30-DEC-85 19:00 | 30-DEC-85 19:00 | 3546 R= | 0 |
| *PRODOS | SYS | 31 | (NO DATE) | 8-DEC-85 22:00 | 15360 | |
| BLOCKS FREE: 203 | BLOCKS USED: 77 | TOTAL BLOCKS: 280 | FILES: 5 | | | |

Mandelbrot Graphics For Commodore

Steven M. Thorpe

A mathematics phenomenon known as the Mandelbrot set can provide the basis for some stunning computer graphics. The following programs make it possible to generate a wide variety of colorful high-resolution images which can be saved on disk for future viewing. The programs work on any Commodore 64 (or 128 in 64 mode).

One of the most beautiful features of the Commodore 64 is its multi-color bitmap mode, which lets you create detailed high-resolution images using several colors. We're all familiar with the many drawing programs that work like video paint boxes, letting you draw directly on the screen. But the computer can create beautiful, highly intricate images all by itself, using relatively simple mathematical methods. "Mandelbrot Graphics For Commodore" allows you to generate interesting hi-res pictures, save them to disk, and reload them at any time.

Creating Screens

Type in and save Program 1. When you run the program, it immediately clears the hi-res screen and begins to draw an image based on the *Mandelbrot set* (see below). You'll need to be patient: A full-screen hi-res image takes a long time to create. Although the program uses machine language routines to clear the hi-res screen, the drawing computations are done in BASIC.

In multicolor bitmap mode, up

to four different colors can be displayed in each character position. Distortion occurs if a program calls for too many colors in a single position. Since you can have only one screen background color at a time, each character position is actually limited to three independent colors. While this may seem a severe restriction, spectacular graphics are still possible. Program 1 selects an available color memory source (not yet used in the current character position unless used for the current color), and then sets the appropriate color code and bit pattern to display that color.

If you're impatient for results and don't mind viewing a smaller image, replace lines 180 and 190 with these lines (be sure you've saved the original version of the program before you make this modification):

```
180 FOR X0=XL TO XR STEP (XR-XL)/  
159*2  
190 FOR Y=YT TO YB STEP (YB-YT)/  
199*2
```

With this change, Program 1 draws an image about one-fourth the size of the screen, in roughly one-fourth the time it would take to draw a full-screen picture. To change back to a full screen, retype lines 180 and 190 as listed in Program 1. This can be useful when modifying the program to produce different results. You can view the results in a reduced scale to save time, then draw it at normal size to be saved to disk. Press RUN/STOP-RESTORE if you need to break out of the program.

The variables XL and YB play a crucial part in defining what the final image looks like. XL sets the left boundary of the Mandelbrot set and YB sets the bottom. Similarly, XR defines the right boundary and YT defines the top. By changing the values of XL, YB, and SR, you can "zoom in" for a closer look at any given area of the Mandelbrot set. You don't need to worry about the value of YT: The program automatically gives YT the value needed to shape the screen image correctly.

Line 170 contains color codes used for various parts of the image. Changing these numbers alters the colors used in the display (see your user's manual for an explanation of color codes). Lines 222-230 shape the zones for each different color. In line 120, the variable SM determines the spacing between different colors, and CT controls how much detail is shown. Remember, using too many colors in zones of rapid color change can lead to excessive color distortion. Together, the variables SM and CT affect how long it takes to complete a Mandelbrot image.

Invisible Lines Of Power

Programs 2 and 3 contain modifications for Program 1 which generate different displays based on the unseen forces of nature. To use these programs, you must already have a copy of Program 1. To enter Program 2, first load Program 1 into memory, then type in the lines listed in Program 2 (they will replace lines 100-240 of the original