ALL ABOUT GRAPHICS

VOLUME I NUMBER 1 *Premiere Issue* SPRING 1983 $4.95

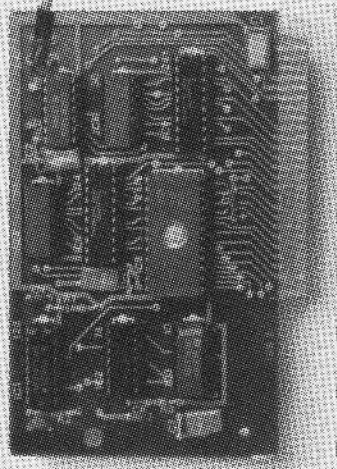
core™

NEW FOR APPLE II

WILDCARD™

MAKES BACK-UP COPIES OF PROTECTED SOFTWARE QUICKLY, EASILY, WITH JUST A PUSH OF A BUTTON.

New software locking schemes have rendered even the latest generation of copy programs virtually unusable. Locksmith™, Nibbles Away™ and other "Nibble copiers" require complicated parameter settings, much patience and great effort to use. More often than not, the results are disappointing. WILDCARD is different. Rather than copying disks track by track, WILDCARD ignores the disk and any copy protection encrypted on it. Instead, WILDCARD takes a snapshot of memory in your Apple® II. Now you can make back-up copies of protected software with the push of a button.



Features

- Hardware copying device... push button operation.
- Copies 48K memory resident software, most 64K software.
- No programming experience or parameters necessary.
- Backs up DOS 3.2 and DOS 3.3 disks.
- Creates DOS 3.3 unprotected and autobooting disks.
- WILDCARD lives in any slot. Undetectable by software.
- Produces autobooting disk in 2 minutes.
- Copies are DOS 3.3 compatible.
- Copies become accessible for alterations.
- Simple, easy-to-use software included.

WILDCARD ***Utility Disk 1*** also available, featuring:

- Automatic program compression and BRUN file maker.
- Multiple programs can be placed on the same disk.
- Recreates basic files to load and save.
- Files can be placed on a hard disk.....and more.

Software is not copy protected. System requirements: Apple II Plus with 64K and DOS 3.3 or Apple *IIe*. Franklin Ace also supported.
*Wildcard does not operate with CP/M® or other microprocessor based software.

Order direct from East Side Software Co., 344 E. 63 St., Suite 14-A, New York City 10021, 212/355-2860. Please include $3.00 for handling. Mail and phone orders may be charged to MasterCard and VISA. N.Y. State residents add sales tax. Dealer inquiries welcome.

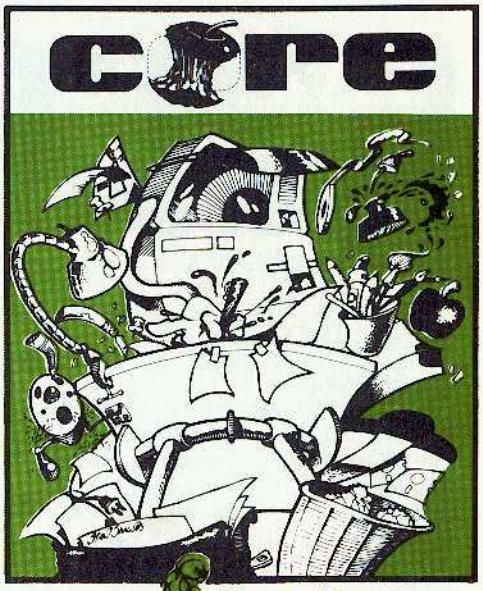
WILDCARD $129.95

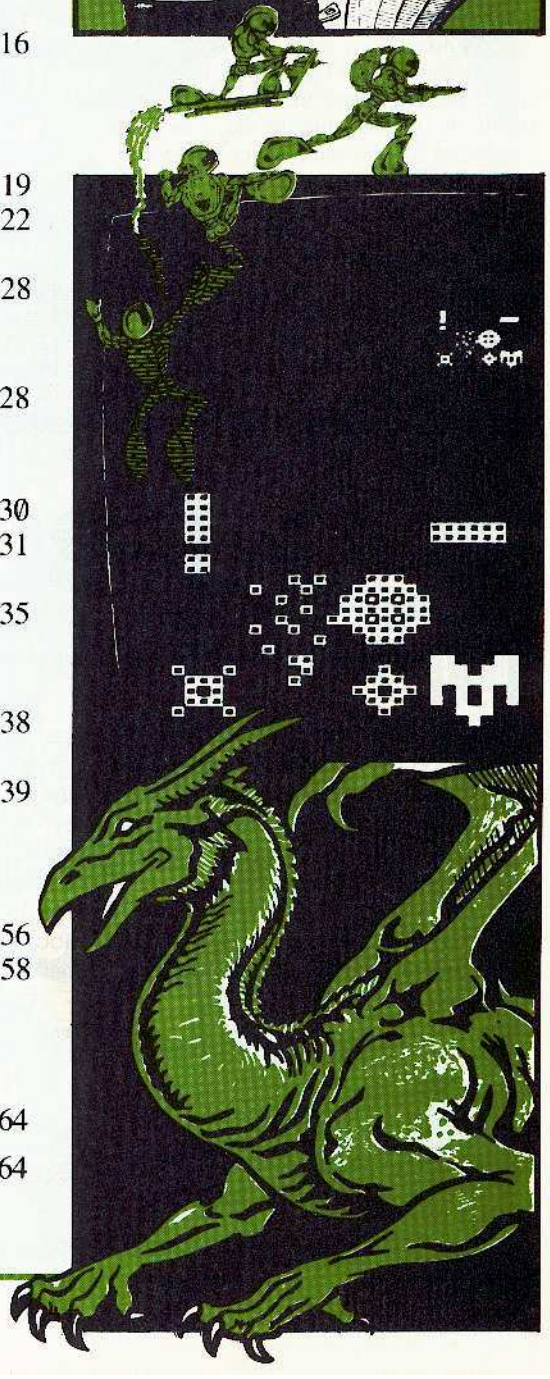
WILDCARD *Utility Disk 1* $30.00

IMPORTANT NOTICE: The WILDCARD is offered for the purpose of enabling you to make archival copies only. Under the Copyright Law you, as the owner of a copy of a computer program, are entitled to make a new copy for archival purposes only and the WILDCARD will enable you to do so. The WILDCARD is offered for no other purpose and you are not permitted to utilize it for any other use, other than that specified.

ALL ABOUT GRAPHICS!

Spring 1983 Premiere Issue Volume I, number 1





Ask and Ye Shall Receive

Recently I had the good fortune of reviewing your magazine (VOL #2 - ISSUE #2). I must say that I was really impressed with the quality of your articles. Your editorials on software copy-protection were informative and very 'up front'. It's about time that a magazine came out and really told it like it is. For this I say 'Keep up the good work'.

Now for the parts I didn't like. First of all is the fact that Hardcore is only published quarterly. I would like to see a monthly magazine for the same cover price. Secondly was the fact that nowhere did I see any information on back issues or back copies of the updates. Are they available and what's the price of these items if available?

That's about all I have to say except, keep the articles interesting and informative and I hope the magazine really prospers.

Daniel L. Masterson
Columbus, OH

Daniel — Starting this month, Softkey will be publishing two magazines, CORE and HARDCORE COMPUTIST, which combined will be a monthly subscription. See the publisher's message on page six for more information.

In addition, to get back copies, see our subscription ad on page four.

Cornelius' Opinions Poor for an Educator

I wish to comment regarding the opinions of Richard Cornelius in "Rebuttal" HardCore Computing 3.0.

In my opinion, this is a very poor attitude ["If each program can be inspected, then those good ideas that people come up with (sic) can be used (pirated) by ayone else"] for an educator to take. Presumably Mr. Cornelius is more concerned with "trade secrets" vis-a-vis the "guild system" rather than the propagation of knowledge. It is incomprehensible and, I believe, inexcusable, for an educator to take this stance.

I have studied the code of several "locked" programs, and if I had written that code, I would prefer that it remained locked also! Please withold my name if you chose to publish this letter.

Name withheld
by request

For readers interested in the debate over copy protection, the upcoming HARDCORE COMPUTIST will contain many letters and articles dealing with the controversial subject.

Publisher:
Charles R. Haight

Editor:
Bev R. Haight

Business and Circulation Manager:
Karen Fitzpatrick

Assistant Editor:
Julie Joringdal

Production:
David C. Smith

Writers and Programmers:
Robb Canfield
Enrique A. Gamez
Rich Orde
Neil Taylor

Illustrators:
Ryuji
Todd Osborne
Luke West

Cover Art:
Stan Shaw

Typeset via Modem:
Graphic Services
Tacoma, WA

Printing:
Grange Printing, Inc.
Seattle, WA

Publishing:
Softkey Publishing
P.O. Box 44549
Tacoma, WA 98444
USA

Address all editorial, advertising, and subscription inquiries to the proper department, CORE, P.O. Box 44549, Tacoma, WA 98444. (For subscription information see page four.) Return postage must accompany all manuscripts, drawings, photos, disks, or tapes if they are to be returned. No responsibility can be assumed for unsolicited manuscripts. We suggest you send only copies.

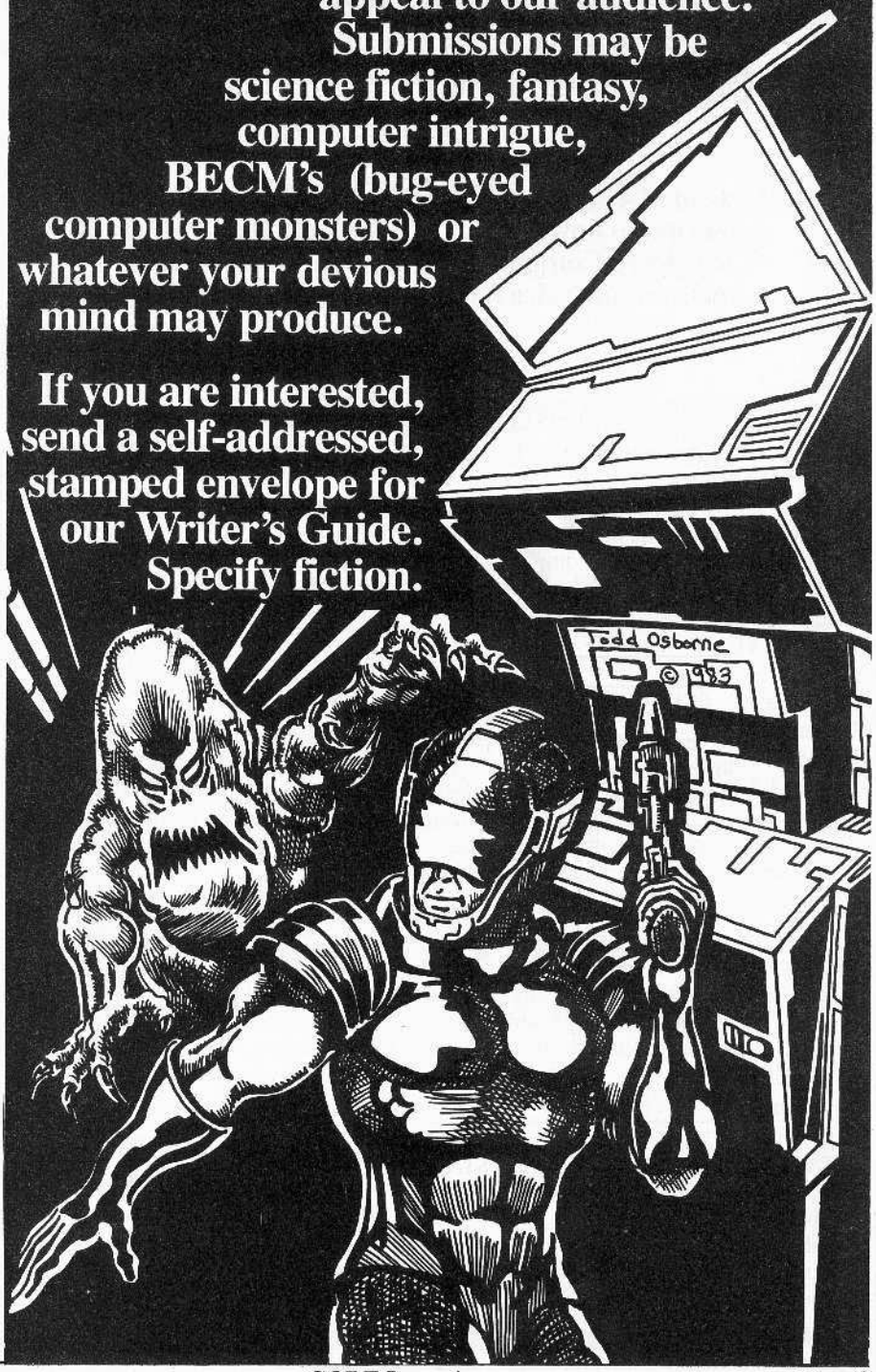
DOMESTIC DEALER RATES sent upon request, or call (206) 581-6038.

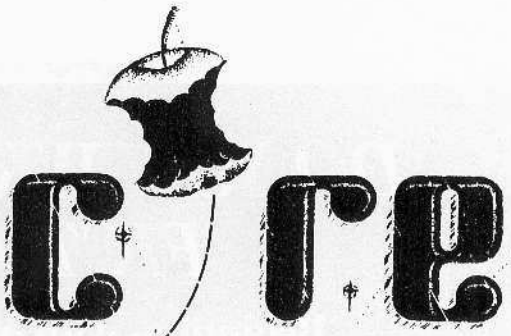
Apple usually refers to the Apple II or II Plus computer and is a trademark of Apple Computers, Inc.

DO YOU WRITE FICTION?

Beginning with our next issue, CORE will publish short stories which will appeal to our audience. Submissions may be science fiction, fantasy, computer intrigue, BECM's (bug-eyed computer monsters) or whatever your devious mind may produce.

If you are interested, send a self-addressed, stamped envelope for our Writer's Guide. Specify fiction.





While the price of everything else in Appledom is shooting SKY-HIGH, subscription rates for the new CORE and HARDCORE COMPUTIST magazines are **HALF** *the cover price!*

Instead of 4 issues and 4 updates (a $20.00 value), a subscription now consists of 12 full-sized magazines (8 of Hardcore Computist and 4 of Core—a ~~$40.00~~ value), yet our price for U.S. subscribers is still

$20.00

Current subscribers: Don't wait to renew. Our rates can't stay this low forever!

[ ] New subscriber
[ ] Renewal

BACK ISSUES:
Hardcore #1 _____
Update 1.1 _____
~~Hardcore #2~~ SOLD OUT!
Update 2.1 _____
Hardcore #3 _____
Update 3.1 _____
Update 3.2 _____

(____) years subscription _____

TOTAL _____

SUBSCRIPTION RATES:

U.S.A.	$20.00
1st Class, APO, FPO	$29.00
Canada	$29.00
Mexico	$32.00
All others	$42.00

BACK ISSUES:
Issues #1,2,3: $5 each, U.S.A.
$8 each, other
Updates: $3 each, U.S.A.
$5 each, other

Subscriptions begin with the current issue. Please specify otherwise. Back issues available only while they last.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Sorry . . .
No phone orders
No credit orders
No purchase orders
U.S. funds only

Make checks payable to:
CORE
P.O. Box 44549
Tacoma, WA 98444

IMPORTANT ANNOUNCEMENT

For those who type in our program listings and are upset about the length of our "Bugs" column, take heart. We're sick of it, too. So, we've taken steps to eliminate the bugs and to make it easy for you to check for your own typographical errors.

First, we will no longer edit program listings. Instead, we will print them just as you would see them on the screen. That means that all print statements will not be neatly formatted, variable names and numbers will be 'wrapped around' . . . in other words, normal Applesoft parsing.

All line numbers will be incremented by ten. That way it will be easy to see if a line is missing.

By making these changes, we should be able to eliminate our "Bugs" column. To help you eliminate your bugs, we will present Checksoft, a program that will check your Applesoft listing to make sure it has been typed in correctly. Checksoft will be featured in the premiere issue of Hardcore Computist.

PROVEN PRODUCTS PRODUCE PROFITS

Our new 4.1 version is by far the most reliable nibble-copy program for the Apple™. There simply *is* no competition. Allows you to backup just about *any* diskette. Includes read/write Nibble Editor, Quickscan Analysis, Media Surface Check, Degauss and Erase, Inspector Interface and Disk-drive Speed Calibration utilities. All for just $99.95 at your local dealer or direct.

Puts all your disk and memory utilities together where they belong — *inside* your Apple. Eprom or disk version is always at your fingertips. Search memory and disks forward and backwards, read nibbles map disk space, locate strings, the uses are endless. At your local dealer or direct — THE INSPECTOR, $59.95.

Includes such goodies as scrolling screen dump, disassembler that shows ASCII, file follower of file, track/sector list-finder by *name*, disk-sector lockout, disk comparer, much more. At your local dealer or direct — WATSON, $49.95. Requires The Inspector.

RAMEX-16™ *(Look Ma, no straps!)* The one 16K memory expansion card for your Apple that *requires no unnecessary surgery.* This board just plugs in with no strap or additional connections. In spite of its quality, the Ramex-16™ costs just $139.95, complete with a *one year* limited warranty.

RAMEX-128™ Adds 128k of additional RAM to your Apple. Used with VisiCalc™ and Super Expander™ to increase work space to 136k. Imagine LOADing a full 136k VisiCalc file into memory in 20 seconds, SAVEing it back in 20 seconds and spending your time working on the template instead of waiting 15-20 minutes as required with other large memory cards. Also included is disk emulation software that adds 8 new commands to DOS as well as super fast LOAD and SAVE of an entire disk. Memory management is also provided to relocate DOS to the card as well as relocating the alternate BASIC. Requires no strapping to the mother board! Works with 16k card as well as alone. And the best part is the low $499.00 price!

SUPER EXPANDER™

is a VisiCalc™ preboot utility that allows up to 136k files using the RAMEX 128. Adds new commands to VisiCalc for super fast (20 second) LOAD and SAVE of files. Just $64.95 — requires RAMEX 128.

A complete turnkey system of memory management on a disk — using either one or two 16K cards. HIDOS™ loads DOS onto one RAM card and with the second card loads an alternate language onto another. SOLIDOS™ turns a 16K card into a fast, 45-sector disk-drive emulator. At your local dealer or direct for just $34.95.

If you use VisiCalc™, then you must have THE CONSOLIDATOR. It will save you hours of keyboard time, by allowing you to manipulate totals of separate files *without reentering them.* Easy to use, invaluable to own. Just $49.95 at your dealer or direct.

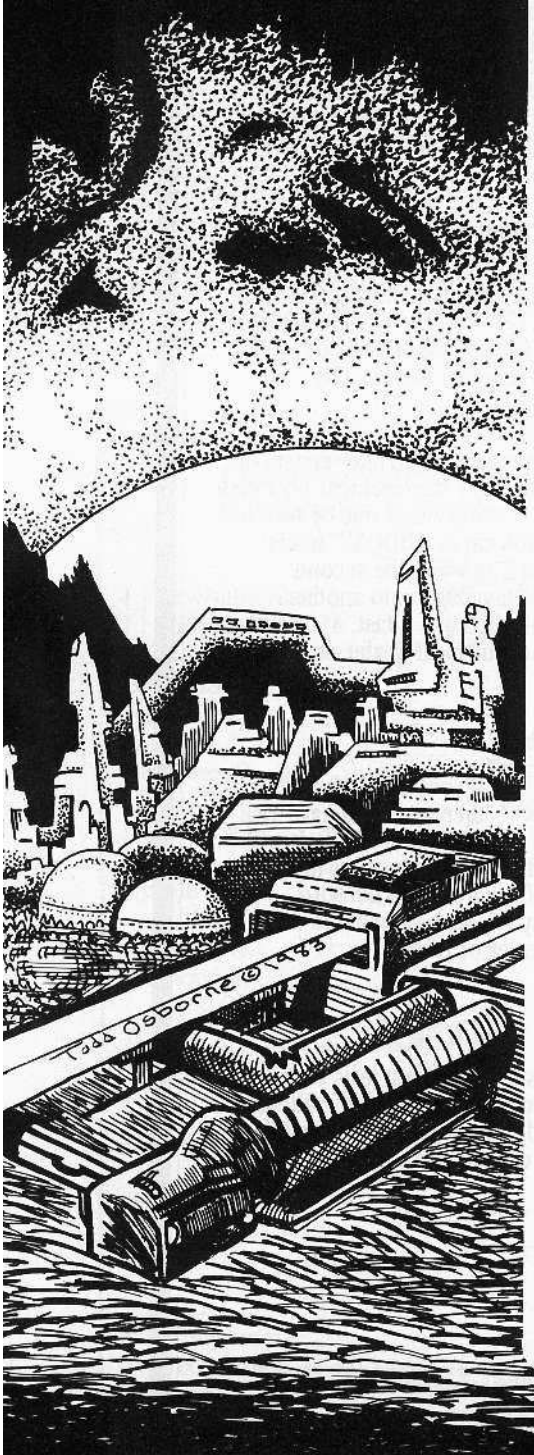
night falls™

is an exciting arcade style game that tests your ability to survive the invasion. Fight off the aliens each night and rebuild your city during the day. See how many nights you can survive.

OMEGA MICROWARE, INC.
222 SO. RIVERSIDE PLAZA
CHICAGO, IL 60606
312-648-4844

Apple is a registered trademark of Apple Computer, Inc.
VisiCalc is a registered trademark of Personal Software, Inc.

this isn't hardcore



Mission:
To boldly go where no Hardcore computist has gone before . . .

Method:
Divide and conquer . . . appledom.

Meaning:
Toto, I don't think we're in the old Hardcore anymore . . .

!click!

. . . THIS IS CORE.
. . . PLEASE ENTER YOUR QUESTION.

Hey. What's going on, here?

. . . THIS IS CORE.
. . . PLEASE ENTER A MORE SPECIFIC QUESTION.

It looks familiar . . .

. . . THIS IS CORE.
. . . PLEASE ENTER QUESTIONS . . .

CORE? It sounds familiar, too.

. . . PLEASE ENTER YOUR QUESTION, NOW.

"All about Graphics" . . . Wasn't HARDCORE #4 supposed to be a special in graphics? The "CORE" part looks the same . . .So where's the 'HARD' part?

. . . IS THAT YOUR QUESTION?

Hmmmm . . . could this be . . . ?

. . . YES, THIS IS CORE.

So, where is HARDCORE COMPUTING #4? You know, the one that's supposed to be a special on graphics?

. . . THIS IS IT.

No, it's not. It just says CORE on the cover. Where's the other half, the "HARD" part? Hey, Hardcore's only half here!

. . . VERY OBSERVANT.
. . . IT IS ONLY HALF HERE.
. . . HARDCORE COMPUTING HAS SPLIT INTO TWO MAGAZINES.
. . . THIS IS CORE.
. . . IT IS A QUARTERLY MAGAZINE THAT WILL TRY TO TACKLE VARIOUS TOPICS OF GREAT INTEREST TO HARDCORE COMPUTISTS.
. . . THIS PREMIERE ISSUE IS:
ALL ABOUT GRAPHICS.
. . . IT WILL BE FOLLOWED BY:
#2, **ALL ABOUT UTILITIES** IN THE SUMMER,
#3, **ALL ABOUT DATA BASES** IN AUTUMN, AND
#4, **ALL ABOUT GAMES** THIS WINTER!
. . . YOU SEE, WE'VE CREATED A MAGAZINE THAT WILL GET RIGHT TO THE VERY CORE OF VARIOUS TOPICS SO THAT . . .

But what happened to the "HARD" part of CORE? I want info on making copies and alterations of all my uncopyable packages.

. . . THE OTHER HALF HAS BECOME THE NEW HARDCORE COMPUTIST, WHICH WILL MAINTAIN THE TRADITIONS OF THE ORIGINAL HARDCORE BY TACKLING THE PROBLEM OF LOCKED-UP SOFTWARE IN A MANNER NOT DONE BEFORE.
. . . REMEMBER THE OLD HARDCORE UPDATES?
. . . WELL, THEY'VE DOUBLED THE PAGES, SLAPPED A SLICK COVER ON IT AND WILL BE ISSUING IT EIGHT TIMES A YEAR.

Does that mean that CORE stands for softcore? Ha, ha!

. . . READ ON, AND FIND OUT . . .
. . . BYE . . .

Hey, I have some more questions . . .
I want to ask if my subscription is going to be applied to both magazines . . .

. . . YES, A SUBSCRIPTION TO HARDCORE COMPUTING MEANS A SUBSCRIPTION TO TWO MAGAZINES:
FOUR ISSUES OF CORE
AND EIGHT ISSUES OF HARDCORE COMPUTIST.
. . . ARE YOU LISTENING?

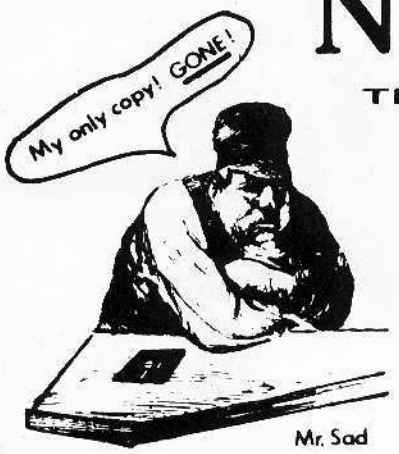
Don't bother me. I'm reading CORE.

. . . SORRY.

Bye . . . .

!click!

NIBBLES AWAY II



Mr. Sad

THE BEST BACK-UP COPY TO DATE

NIBBLES AWAY II, second edition, is a greatly enhanced and improved version of our earlier product NIBBLES AWAY. Many new and exciting features have been implemented in **NIBBLES AWAY II** to insure 'State of the Art' reliability.
Other similar systems on the market today (regardless of price) can't begin to compare with all the features built into '**NIBBLES AWAY II**".

The following summary will introduce you to some of the many new, and asked for features, incorporated in our new release...

AGAIN! Ahead of all others.

AUTO-LOAD PARAMETERS... Free's the user from having to Manually Key In Parm values used with the more popular software packages available for the Apple II.

VISA MasterCard APPLE 48K

A Screen Display of the 'Auto-load' Parameters actually loading, permits the user to STUDY the changes being made to the system!

EXPANDED USER MANUAL .. Incorporates new Tutorials for all levels of expertice; Beginners Flowchart for 'where do I begin' to 'Advanced Disk Analysis' is included.

TRACK/SECTOR EDITOR..... An all new Track/Sector Editor, including the following features: Read, Write, Insert, Delete Search, and impressive Print capabilities!

DISK DIAGNOSTICS..... Checks such things as: Drive Speed, Diskette Media Reliability, and Erasing Diskettes.

PROGRAM MODIFICATION... An all new way to deal with Parameter changes; NA-II displays them On Screen, and in a Text format that is as easy to use as pressing one key to move through the listing!

FAST & MORE ACCURATE ... A FAST program with ACCURATE error detection makes '**NIBBLES AWAY II**' a genuinely superior product compared with others. **Rated fourth by SOFTALK readers** (8.25 **out of** 10)

UPDATE TO NA-II ..... Updates will be made available to all earlier versions through COMPUTER: applications for a modest charge of Ten Dollars, to cover manual, Diskette, and Postage & Handling. Please include NAME and SERIAL # of ORIGINAL NIBBLES AWAY.

LIST ONLY ~~$69.95~~

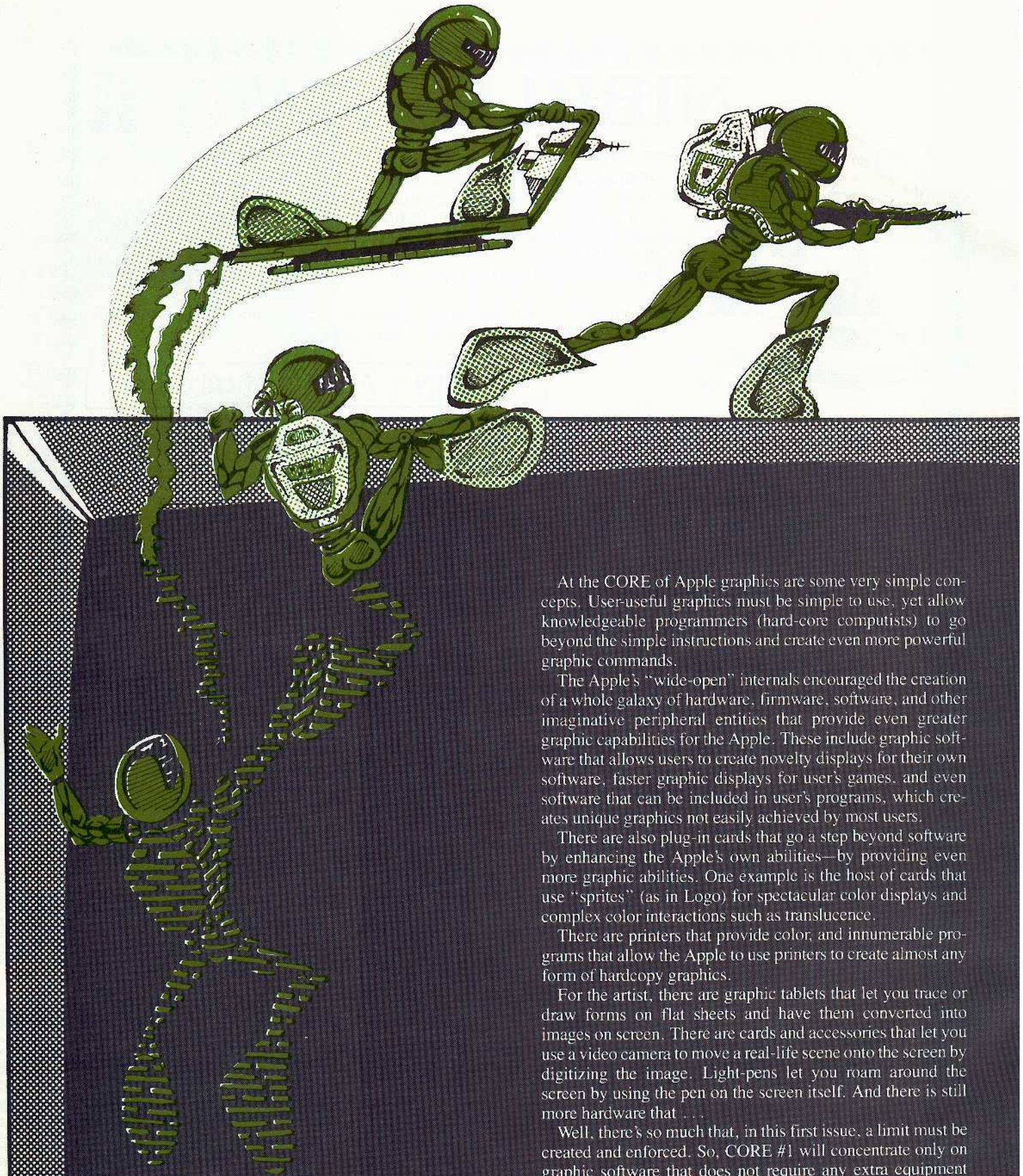
Special HARDCORE $ **$60**

201-838-6463

DISCOUNT COMPUTER PRODUCTS

Mr Happy

PO BOX 31
RIVERDALE, N J
07457



At the CORE of Apple graphics are some very simple concepts. User-useful graphics must be simple to use, yet allow knowledgeable programmers (hard-core computists) to go beyond the simple instructions and create even more powerful graphic commands.

The Apple's "wide-open" internals encouraged the creation of a whole galaxy of hardware, firmware, software, and other imaginative peripheral entities that provide even greater graphic capabilities for the Apple. These include graphic software that allows users to create novelty displays for their own software, faster graphic displays for user's games, and even software that can be included in user's programs, which creates unique graphics not easily achieved by most users.

There are also plug-in cards that go a step beyond software by enhancing the Apple's own abilities—by providing even more graphic abilities. One example is the host of cards that use "sprites" (as in Logo) for spectacular color displays and complex color interactions such as translucence.

There are printers that provide color, and innumerable programs that allow the Apple to use printers to create almost any form of hardcopy graphics.

For the artist, there are graphic tablets that let you trace or draw forms on flat sheets and have them converted into images on screen. There are cards and accessories that let you use a video camera to move a real-life scene onto the screen by digitizing the image. Light-pens let you roam around the screen by using the pen on the screen itself. And there is still more hardware that . . .

Well, there's so much that, in this first issue, a limit must be created and enforced. So, CORE #1 will concentrate only on graphic software that does not require any extra equipment outside of a regular 48K Apple II (with Applesoft in ROM) unless otherwise specified.

Even with this limitation, we are left with a huge field of software and information that cannot possibly be condensed to our magazine size and format. So what we have included in this premiere issue will cover all the capabilities of the native Apple, unadorned and unenhanced.

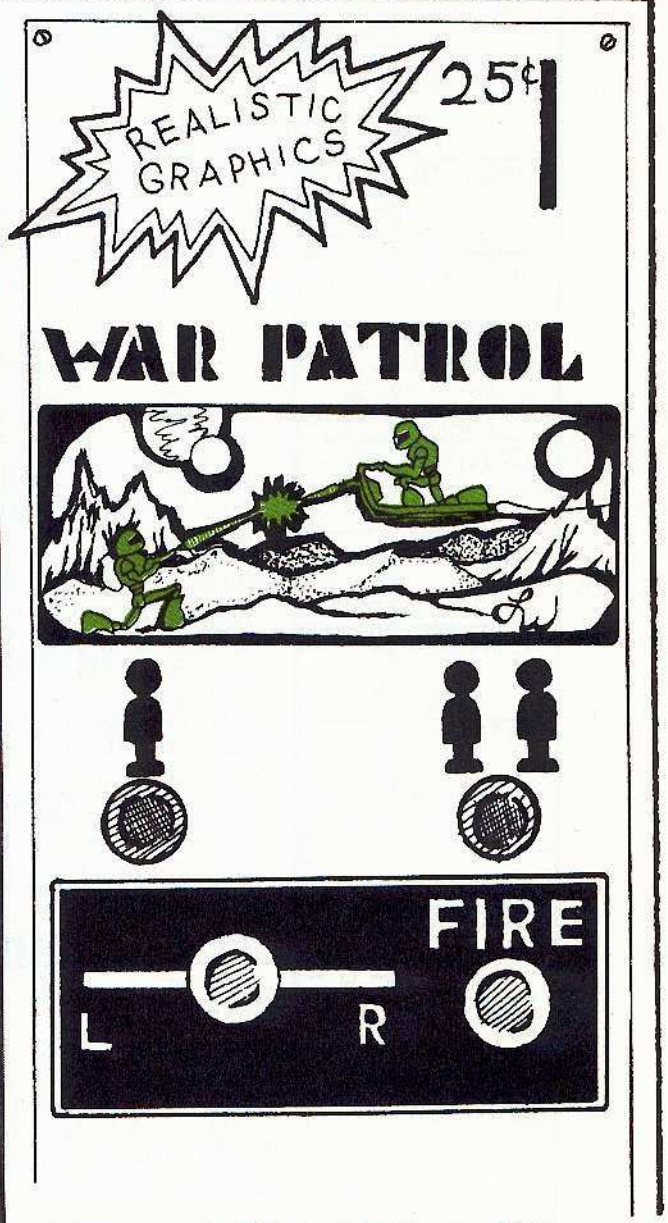
In order to cover all the capabilities of the naked Apple, we've divided its abilities and uses into a few broad categories. In the order that we present them, these categories are:

1. **TEXT PAGE GRAPHICS**
2. **LO-RESOLUTION GRAPHICS**
3. **HI-RESOLUTION GRAPHICS**
4. **COLOR**
5. **VECTOR GRAPHICS (SHAPE TABLES)**
6. **BLOCK GRAPHICS**
7. **ANIMATION**

Included are useful graphic utilities and short but interesting programs. A special feature in this issue is Canfield's character generator, Quick Draw. Some of the interesting programs include a game that uses the block/character generator, a short shape table editor, a fascinating demonstration of shape table graphics, and other "demo" programs.

If we missed anything, we'll be sure to include them in CORE's own column (MORE FROM CORE) in the following issues of the new HARDCORE COMPUTIST.

Note: For those who frown on the idea of typing in all the programs from this issue, you'll be happy to know that the major programs can be found on a disk available through Softkey Publishing. The disk will retail for $24.95, but Hardcore Computing subscribers can purchase it at the reduced rate of $19.95.



MEMORY MAP

The most important aspect to remember about computer graphics is that everything displayed on the screen can be explained in terms of memory. When you look at text page graphics you are really staring at a piece of memory displayed on the screen as text. In the same way, when you watch the multi-hued low resolution (lo-res) display, you are staring at that same piece of memory displayed in yet another way. Of course, that means that all high resolution pictures (hi-res) are pictures of memory (oddly formatted I admit, but only memory).

With that simple fact in mind, the very first subject that must be fully understood is Apple memory.

Whenever memory format is discussed or explained in CORE, it will probably be accompanied by a "memory map". In every instance, this bare map will have memory divided into "pages" consisting of 256 bytes. It's best to examine the memory map in order to see these divisions. There are four pages to every "K" (kilobyte, but pronounced "kay"). Don't confuse these pages with the "pages" used to describe the various graphic display buffers (such as text page one and two, or hi-res page one and two). On this bare map you will also find important "memory landmarks", such as the buffers described above.

Because programs use memory in different ways, a firm visual "feel" for memory layout is important to the understanding of how and why various programs work the way they do. The memory map will be illustrated in order to show how memory is being used by the various programs.

The "normal 48K" Apple has a total of 256 pages of memory. That's 65,536 bytes, which is 64K. The 48K used to describe the Apple indicates the total RAM (Random Access Memory) it has. The other 16K is in ROM (Read-Only Memory). Users can only put programs into the RAM, but not all of RAM is available. This is especially evident if you use a disk drive. The Disk Operating System (DOS, pronounced "dose" or "doss") program occupies much of RAM (see Memory Map).

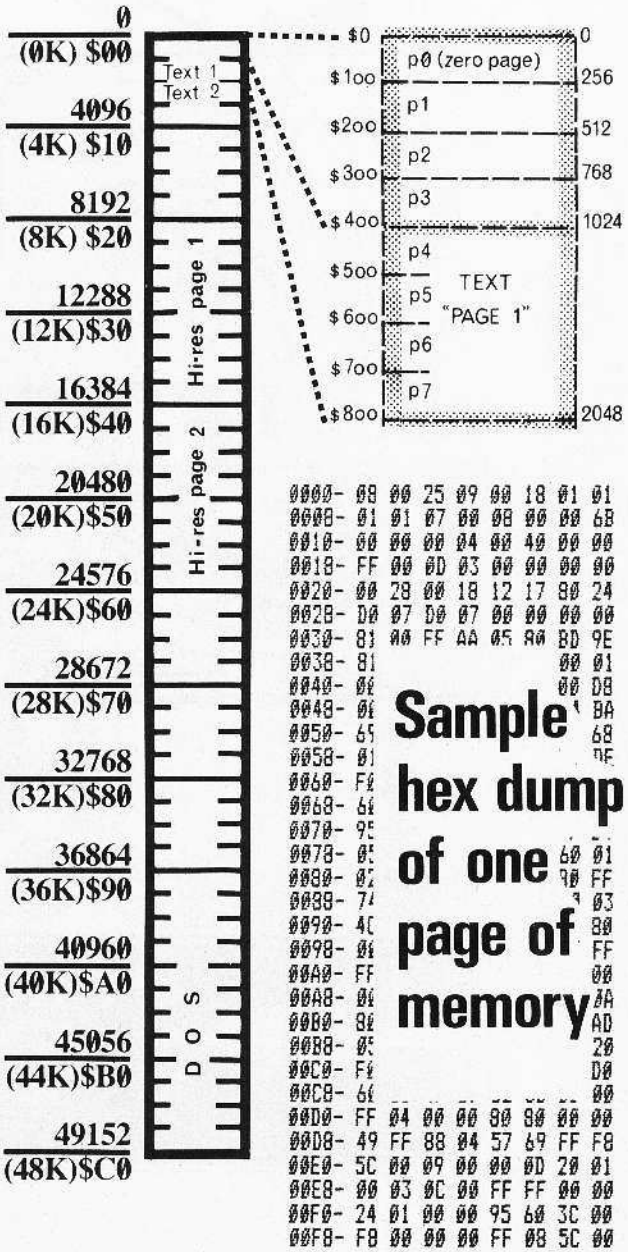
And if you have less than 48K, or you have to load Applesoft into memory, you may discover that your graphics ability has been dramatically reduced. This is because, as mentioned earlier, graphics display is memory display, and if that memory is used for something else it's not available for graphics.

All the programs in this issue require 48K with Applesoft in ROM, leaving RAM almost totally free for optimal graphic utilization—almost . . .

Parts of the first three pages of RAM memory are used by the monitor, Applesoft or DOS, although there are unused gaps that come in handy when you need someplace to tuck a few small machine routines. There will be much more on memory in the next issue of CORE (the Utilities special issue).

More detail on memory will be explained at appropriate sections of this issue.

Landmarks



NOT COPY-PROTECTED

GraFORTH®

by Paul Lutus

Make Your Graphics Come Alive!

GraFORTH combines sophisticated graphics features with a powerful programming language. Much more than a utility program, GraFORTH's superior graphics make it the ultimate language for entertainment and educational software creation. Included are plotting and line graphics, text display and character image graphics, and high speed 3-D graphics, all with a variety of colors and drawing options. GraFORTH can be used on a 48K Apple][system with DOS 3.3 and one disk drive. A 16K memory card is a useful option.

The Language:

GraFORTH is a graphics language similar in structure to FORTH, but entirely rewritten for ease of use and maximum speed. (Counting to 30,000 in GraFORTH takes less than three seconds!) Immediate commands and programs can be entered and run directly from the keyboard. GraFORTH includes a full set of arithmetic and string handling capabilities. Since standard DOS files are used, communication with other programs and languages is straightforward.

Plotting and Line Graphics:

The first level of graphics consists of plotting points, drawing lines, and filling areas in any of the Apple's high-resolution colors. Lines are drawn much faster than in Basic, and colored lines are never broken. Lines and areas can also be neatly erased from the screen without disturbing other images. Turtle-graphics are included to draw line shapes rapidly at any angle.

Text Display and Character Graphics:

GraFORTH displays both upper and lower case characters. You can use any of the five character sets provided, or create your own with the character editor. Character shapes may be combined to form a single multi-character image, then "block printed" at high speed anywhere on the Apple screen. Characters and character shapes can be drawn in color up to 8 times their normal size.

Three-Dimensional Graphics:

GraFORTH can also draw three-dimensional color images at speeds that make animation possible! Up to sixteen 3-D objects can be manipulated simultaneously. Images can be rotated, scaled, translated, and positioned, with or without perspective. The supplied image editor allows you to create your own 3-D images. Colors may be specified as an image is created, or selected when the image is drawn.

Music:

GraFORTH includes a sophisticated software-based music synthesizer for adding music or sound effects to your programs. Music can be played in any one of several instrument voices.

The System:

Programs written in GraFORTH can be saved to disk as complete stand-alone systems that do not require any additional software to run. This makes GraFORTH the ideal language for developing games and other graphics software.

The Package:

GraFORTH is supplied on a diskette with a special version of DOS 3.3 that loads into a language card (if present), freeing up more memory for your programs. The disk includes many sample image files, utilities, and complete demonstration programs detailing the features of GraFORTH. Included with the disk is a 200-plus page tutorial explaining the ins and outs of GraFORTH. No previous programming experience is necessary to use GraFORTH.

Order GraFORTH from Computer Hideout today!

COMPUTER HIDEOUT

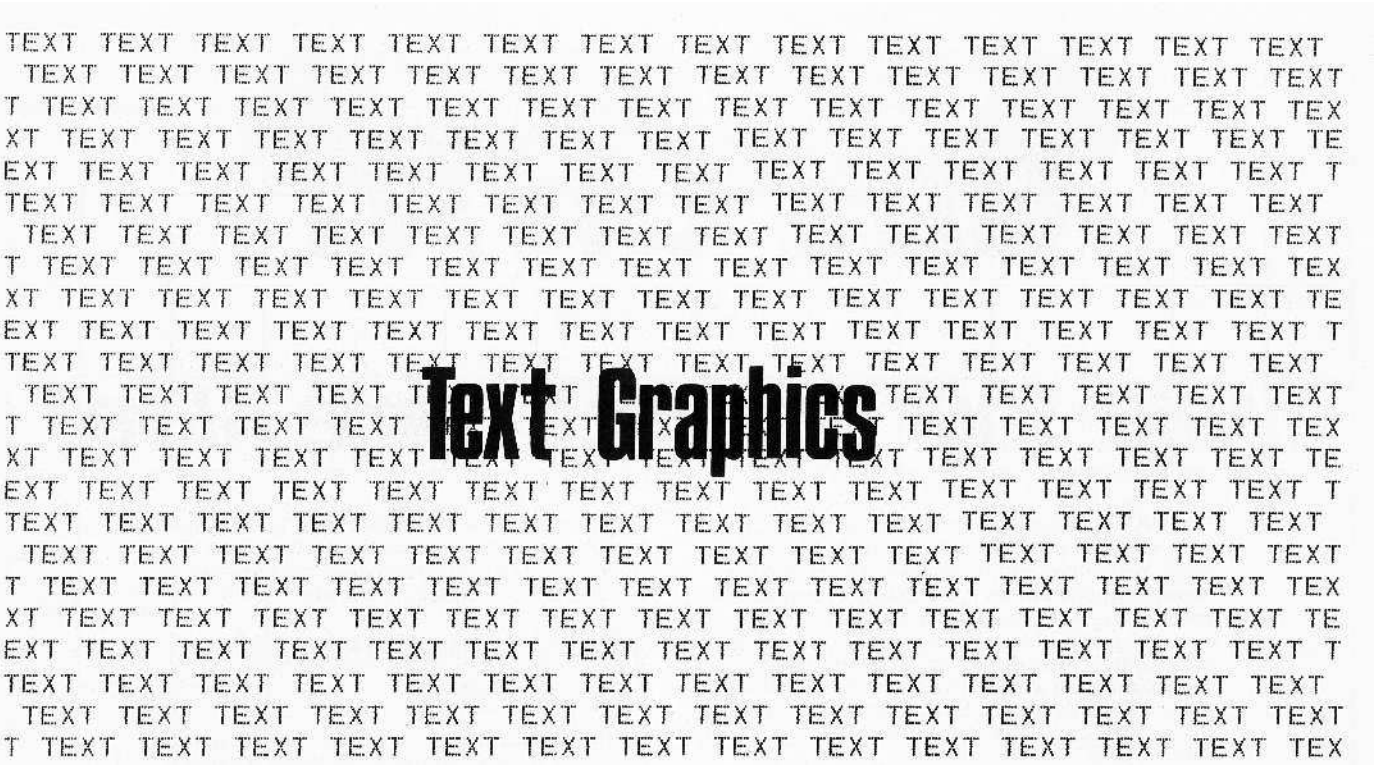
INTELLIGENT PRODUCTS FOR MICROCOMPUTERS

TERMS: We Accept M.C. or VISA (Indicate Card # and Expiration Date) Personal Checks - Allow 3 weeks to clear. Please Enclose $2.00 for Postage and Handling. Louisiana Residents - Please Add 6% Sales Tax.

$64.95

(318) 942-9446

P. O. Box 264 / Opelousas, Louisiana 70570



Contrary to what some hardcore computists claim, it is neither unsophisticated nor infantile to use the text page for graphic displays. It is true, however, that some computists ignore the text page because they are adept at using the high resolution pages for graphics. For simple (colorless?) graphics, the text page should be utilized because it is so easy to use. In fact, many novice programmers use the text page for arcade games .... And, of course, all text-only adventure games use this page because it allows the program to occupy all the rest of memory (which is needed by even the simplest adventure requiring a lot of text).

The text "screen" consists of 24 horizontal lines (rows) with 40 character spaces (columns) on each line. The "character" set that can be placed on this image area (the text page) consists of the ten digits, 26 upper case letters and 28 special characters (like ?!*= etc.) for a grand (ha,ha) total of 64 different graphic symbols that can be in NORMAL (white on a black background), INVERSE, (black on white background) and FLASH (flashing between inverse and normal).

(Note: The Apple //e has lower case letters as well.)

Each character is composed of a matrix of seven pixels across by eight pixels high. The letters themselves are five pixels across by seven pixels high, leaving blank one vertical row of pixels on each side and one blank horizontal row on the bottom.

There are two text "pages" occupying consecutive positions in memory. Each page consists of one kilobyte (K) or 1024 bytes. However, 40 columns by 24 rows is 960 bytes, which means 64 bytes are not displayed. The memory for text page one is from 1024 ($400) to 2047 ($7FF). Page two is from 2048 ($800) to 3071 ($BFF).

(Note: Normally, your Applesoft program begins at 2048 ($800), which means that your program is on page two of text.)

Horizontally, each set of 40 columns represents 40 consecutive bytes in memory. Vertically, however, the pattern is broken. If the text screen was split into three equal horizontal sections of eight rows each and the sections were placed side by side, the rows would then be consecutive.

From the point of view of memory, the text page is similarly broken, jumping to three separate locations every 40 bytes.

This information is valuable if you want to use text page two for display purposes. Remember, if you use text page two for graphics your Applesoft program must begin above 3071 ($BFF), or you will be writing over the program.

Graphics for the text page is almost always PRINTed to the screen. Exact positions can be specified using:

PRINT TAB (?) "?"	**PRINT SPC (?) "?"**
HTAB ?	**POKE 36, ?**
VTAB ?	**POKE 37, ?**

Other text graphic commands include:

TEXT	**ctrl-M**
HOME	**X = PEEK (36)**
CALL-936	**X = PEEK (37)**
ctrl-J	**X = POS (0)**
CALL-922	**FLASH**
CALL-958	**INVERSE**
CALL-868	**NORMAL**
CALL-912	

The TAB command lets you "tab" over to a specific text column. TAB (1) is the left-most column while TAB(40) is the right-most column. TAB must be used within a PRINT statement. For example:

PRINT TAB(10)"CORE"

This will print the word "core" starting at column 10. Remember, TAB will "tab" only to the right, never to the left, of the current line.

The SPC command lets you put spaces within a PRINT statement without having to put spaces within the print's quotes.

PRINT "HELLO THERE"

can be replaced by

PRINT "HELLO" SPC(13)"THERE"

Outside a PRINT statement you must use HTAB.

```
PRINT "HELLO";
HTAB 18
PRINT "THERE"
```

HTAB, like TAB, allows printing to begin at specific columns, no matter where the last statement was printed. SPC puts spaces between statements and depends upon the character previously printed.

Unlike TAB, HTAB can move the cursor to the left.

VTAB acts like HTAB, except that it specifies the row (vertical position) from which printing begins.

All VTABs must be in the range of one through 24, with the top row as one and the bottom row as 24. VTAB can move the cursor both up and down.

The acceptable values for TAB and SPC include zero through 255, but the left-most position is specified by one. TAB(41) also specifies the right-most position but on the next line down, followed by 81, 121, 161, 201, and 241. Similarly, the right-most position is specified by TAB(40), followed by TAB(80) for the next line down, and then by 120, 160, 200, and 240. HTAB works the same way.

PRINT TAB (40) or PRINT SPC (41) will clear a line of text or, after specifying INVERSE, will create an inverse line.

This means that you can INVERSE or FLASH the entire text image area by:

```
PRINT TAB(240)TAB(240)TAB(240)TAB(240)
VTAB 1
```

The same goal can be accomplished with:

```
PRINT SPC(241)SPC(241)SPC(241)SPC(241)
VTAB 1
```

(The VTAB 1 will stop the screen from scrolling by placing the cursor at the top of the text screen.)

The cursor position can be controlled by using POKE 36 and POKE 37. In other words, POKE 36,0 works like HTAB 1. And POKE 37,0 is like VTAB 1. The values to POKE into 36 should be limited to a range from zero to 39, while the values to POKE into 37 should be confined to zero through 23. Remember, the value used in these POKEs is one less than the value you use in a TAB, HTAB, or VTAB.

The present position of the cursor can be determined by PEEKing 36 and 37.

You can also use POS to find the current horizontal POSition of the cursor. Like PEEK (36), you will get a value from zero to 255.

When you need emphasis within the text, you can specify INVERSE or FLASH, but remember to follow the print statement with NORMAL.

For a demonstration of some of these commands, type in this small program:

```
0 REM JAGGED SCROLLER
1 INVERSE : VTAB 24:A1 = 1:A2 = 20:ST = 1
2 FOR A = A1 TO A2 STEP ST: HTAB A: PRINT  SPC( 20):
    PRINT : NEXT :A0 = A1:A1 = A2:A2 = A0:ST =  - S
  T: GOTO 2
```

A more elaborate way to control the text screen is to specify the dimensions of the "text window." Normally set by TEXT to 40 columns by 24 rows, the window's dimensions can be altered by POKEing four addresses with appropriate values.

Window Dimension	Address to Poke	Minimum Value	Maximum Value	Text Default
left	32	0	39	39
width	33	1	40	40
top	34	0	24	24
bottom	35	0	23	23

WARNING: Do not set the dimensions outside the minimum-maximum range. For example, POKE 33,0 (which is below the minimum) will "bomb" Applesoft by over-writing some valuable Applesoft data.

Some commands affect only areas within the window:

HOME (CALL -936) will erase everything inside a window and puts the cursor at the window's upper left corner.

ctrl-J (CALL -922) moves the cursor down one line.

CALL -912 scrolls the window upward, erasing the uppermost line of text and placing a new line at the bottom.

CALL -958 erases everything below and to the right of the cursor.

CALL -868 erases everything just to the right of the cursor.

The TEXT command resets the window's dimensions to the default values.

Most utilities and business software use the text page to display viewer information. Rarely is this text actually used in a creative manner because it requires extra code or would detract from the actual usefulness (ease of viewing) of the information being displayed.

However, there are ways to express text information creatively using just the text page. These include the often-used inverse and flashing modes, and an excess of asterisks and dashes . . . but they sometimes include walking letters, falling letters, marquee effects, wobbly lines, moving arrows, and various boxes that shrink, enlarge, or fragment.

Those who recall *Text Invaders (HardCore Computing #2)* realize that even games can be expressed on the text page. And *Zephyr Wars (HardCore Computing #3)* showed how the text page can be flashed so that text appears to be on the hi-res page.

continued on page 54

Low Resolution Graphics

From text to low resolution (lo-res) graphics might seem a big step (from letters to color blocks), but it's not. You're still "looking" at the same block of memory (text page), except that now each byte is displayed as two blocks of color stacked one on top of the other. The two blocks take up the same space on the screen as does one character, but because there are twice as many blocks as there are characters, the lo-res screen is an array of 40 across and 48 high. Each block can be any of 16 colors, depending upon the value of the two nybbles that make up the byte. The lower color is determined by the "high" nybble and the upper color by the "low" nybble.

Bottom Color = INT (BYTE/16)
Top Color = BYTE—BOTTOM COLOR * 16

Use the Color Code to discover the code for each color. You can avoid the mathematics involved in determining the high and low nybble by using the Low Resolution Color Character Chart. It gives both the top and bottom colors for all 256 different byte values, as well as the character that is displayed on the text screen. That's right . . . by switching to text, the lo-res colors are revealed as ASCII characters. The color character chart is most useful in determining the relationship of color to text. This knowledge comes in handy when using hybrid graphic commands, such as lo-res commands for text graphics. (See Shrinking and Enlarging Boxes.)

There are two low resolution display modes: full screen and mixed screen. With mixed screen the bottom four rows of text are displayed while the rest is in lo-res colors. Like text graphics, you can use page one and page two. However, most Applesoft programs begin on page two of text.

GR initiates lo-res graphics by displaying page one in the mixed screen mode. To switch from page one to page two, POKE -16299,0. To return to page one, POKE -16300,0. To initiate lo-res graphics without GR, POKE -16304,0 if you are on the text page or POKE -16298,0 if you are in high resolution. To switch to mixed screen, POKE -16301,0. For full screen, POKE -16302,0.

Once in lo-res, the following commands will put color on the screen:

COLOR = ?
PLOT X,Y
HLIN Y1,Y2 at X
VLIN X1,X2 at Y

Color is set to zero by the GR command. Other colors must be specified by setting color from zero to 255 because the value placed after COLOR = is always divided by 16 and the remainder used to select a color. Setting color to any other number is an ILLEGAL QUANTITY ERROR.

Once the color has been selected, the position must be given. The horizontal position (X) and its vertical block position (Y) must follow the PLOT command. X must be in the range of zero through 39, where zero is the far left edge. Y must be between zero and 47; zero is at the top of the screen. Any value outside these limits will be an ILLEGAL QUANTITY ERROR.

Using PLOT while in the TEXT mode will put a character on the text screen. For example:

COLOR = 1: PLOT 0,1

The result will be an inverse 'P' in the upper left part of the screen.

Low Resolution Color Character Chart (Hex and Decimal)

Mode: INVERSE
Bottom Color: 0

DEC	HEX	TOP	ASC
0	00	0	@
1	01	1	A
2	02	2	B
3	03	3	C
4	04	4	D
5	05	5	E
6	06	6	F
7	07	7	G
8	08	8	H
9	09	9	I
10	0A	10	J
11	0B	11	K
12	0C	12	L
13	0D	13	M
14	0E	14	N
15	0F	15	O

Mode: INVERSE
Bottom Color: 1

DEC	HEX	TOP	ASC
16	10	0	P
17	11	1	Q
18	12	2	R
19	13	3	S
20	14	4	T
21	15	5	U
22	16	6	V
23	17	7	W
24	18	8	X
25	19	9	Y
26	1A	10	Z
27	1B	11	[
28	1C	12	\
29	1D	13	]
30	1E	14	^
31	1F	15	_

Mode: INVERSE
Bottom Color: 2

DEC	HEX	TOP	ASC
32	20	0	
33	21	1	!
34	22	2	"
35	23	3	#
36	24	4	$
37	25	5	%
38	26	6	&
39	27	7	'
40	28	8	(
41	29	9	)
42	2A	10	*
43	2B	11	+
44	2C	12	,
45	2D	13	-
46	2E	14	.
47	2F	15	/

Mode: INVERSE
Bottom Color: 3

DEC	HEX	TOP	ASC
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	10	:
59	3B	11	;
60	3C	12	<
61	3D	13	=
62	3E	14	>
63	3F	15	?

Mode: FLASH
Bottom Color: 4

DEC	HEX	TOP	ASC
64	40	0	@
65	41	1	A
66	42	2	B
67	43	3	C
68	44	4	D
69	45	5	E
70	46	6	F
71	47	7	G
72	48	8	H
73	49	9	I
74	4A	10	J
75	4B	11	K
76	4C	12	L
77	4D	13	M
78	4E	14	N
79	4F	15	O

Mode: FLASH
Bottom Color: 5

DEC	HEX	TOP	ASC
80	50	0	P
81	51	1	Q
82	52	2	R
83	53	3	S
84	54	4	T
85	55	5	U
86	56	6	V
87	57	7	W
88	59	8	X
89	59	9	Y
90	5A	10	Z
91	5B	11	[
92	5C	12	\
93	5D	13	]
94	5E	14	^
95	5F	15	_

Mode: FLASH
Bottom Color: 6

DEC	HEX	TOP	ASC
96	60	0	
97	61	1	!
98	62	2	"
99	63	3	#
100	64	4	$
101	65	5	%
102	66	6	&
103	67	7	'
104	68	8	(
105	69	9	)
106	6A	10	*
107	6B	11	+
108	6C	12	,
109	6D	13	-
110	6E	14	.
111	6F	15	/

Mode: FLASH
Bottom Color: 7

DEC	HEX	TOP	ASC
112	70	0	0
113	71	1	1
114	72	2	2
115	73	3	3
116	74	4	4
117	75	5	5
118	76	6	6
119	77	7	7
120	78	8	8
121	79	9	9
122	7A	10	:
123	7B	11	;
124	7C	12	<
125	7D	13	=
126	7E	14	>
127	7F	15	?

Mode: CONTROL
Bottom Color: 8

DEC	HEX	TOP	ASC
128	80	0	@
129	81	1	A
130	82	2	B
131	83	3	C
132	84	4	D
133	85	5	E
134	86	6	F
135	87	7	G
136	88	8	H
137	89	9	I
138	8A	10	J
139	8B	11	K
140	8C	12	L
141	8D	13	M
142	8E	14	N
143	8F	15	O

Mode: CONTROL
Bottom Color: 9

DEC	HEX	TOP	ASC
144	90	0	P
145	91	1	Q
146	92	2	R
147	93	3	S
148	94	4	T
149	95	5	U
150	96	6	V
151	97	7	W
152	98	8	X
153	99	9	Y
154	9A	10	Z
155	9B	11	[
156	9C	12	\
157	9D	13	]
158	9E	14	^
159	9F	15	_

Mode: NORMAL
Bottom Color: 10

DEC	HEX	TOP	ASC
160	A0	0	
161	A1	1	!
162	A2	2	"
163	A3	3	#
164	A4	4	$
165	A5	5	%
166	A6	6	&
167	A7	7	'
168	A8	8	(
169	A9	9	)
170	AA	10	*
171	AB	11	+
172	AC	12	,
173	AD	13	-
174	AE	14	.
175	AF	15	/

Mode: NORMAL
Bottom Color: 11

DEC	HEX	TOP	ASC
176	B0	0	0
177	B1	1	1
178	B2	2	2
179	B3	3	3
180	B4	4	4
181	B5	5	5
182	B6	6	6
183	B7	7	7
184	B8	8	8
185	B9	9	9
186	BA	10	:
187	BB	11	;
188	BC	12	<
189	BD	13	=
190	BE	14	>
191	BF	15	?

Mode: NORMAL
Bottom Color: 12

DEC	HEX	TOP	ASC
192	C0	0	@
193	C1	1	A
194	C2	2	B
195	C3	3	C
196	C4	4	D
197	C5	5	E
198	C6	6	F
199	C7	7	G
200	C8	8	H
201	C9	9	I
202	CA	10	J
203	CB	11	K
204	CC	12	L
205	CD	13	M
206	CE	14	N
207	CF	15	O

Mode: NORMAL
Bottom Color: 13

DEC	HEX	TOP	ASC
208	D0	0	P
209	D1	1	Q
210	D2	2	R
211	D3	3	S
212	D4	4	T
213	D5	5	U
214	D6	6	V
215	D7	7	W
216	D8	8	X
217	D9	9	Y
218	DA	10	Z
219	DB	11	[
220	DC	12	\
221	DD	13	]
222	DE	14	^
223	DF	15	_

Mode: L. CASE
Bottom Color: 14

DEC	HEX	TOP	ASC
224	E0	0	
225	E1	1	!
226	E2	2	"
227	E3	3	#
228	E4	4	$
229	E5	5	%
230	E6	6	&
231	E7	7	'
232	E8	8	(
233	E9	9	)
234	EA	10	*
235	EB	11	+
236	EC	12	,
237	ED	13	-
238	EE	14	.
239	EF	15	/

Mode: L. CASE
Bottom Color: 15

DEC	HEX	TOP	ASC
240	F0	0	0
241	F1	1	1
242	F2	2	2
243	F3	3	3
244	F4	4	4
245	F5	5	5
246	F6	6	6
247	F7	7	7
248	F8	8	8
249	F9	9	9
250	FA	10	:
251	FB	11	;
252	FC	12	<
253	FD	13	=
254	FE	14	>
255	FF	15	?

There are also commands that let you draw horizontal and vertical lines. Once the color is specified, HLIN draws horizontal lines. You must specify the column to begin drawing (Y1) and the column to stop drawing (Y2), followed by the horizontal row (X) on which to draw this line. The format of the command is:

HLIN Y1, Y2 AT X

(The comma separating the start and end columns is necessary, as well as the word "at" which indicates the proper row.)

VLIN works just like HLIN, except the beginning (X1) and end (X2) rows must be specified along with the column (Y) upon which the line will be drawn. In other words:

VLIN X1, X2 AT Y

If X is set less than zero or greater than 39, or if Y is less than zero or greater than 47, you will get an ILLEGAL QUANTITY ERROR.

Like PLOT, both HLIN and VLIN will put characters on the text page.

The color of any block on the lo-res screen can be determined by using the SCRN (X,Y) command. A number from zero to 15 will be returned. Use the lo-res color chart to get the color for that number. The values for X and Y must be between zero and 47. Values greater than 47 but less than 255 will still work, but the numbers returned are not related to the lo-res screen.

continued on page 55

COLOR CODE

Ø	Black	8	Brown
1	Magenta	9	Orange
2	Dark Blue	1Ø	Grey 2
3	Purple	11	Pink
4	Dark Green	12	Light Green
5	Grey 1	13	Yellow
6	Medium Blue	14	Aquamarine
7	Light Blue	15	White

What Characters are also Solid Lo-Res Colors?

(Top and bottom colors the same.)

DEC	HEX	ASC	MODE	COLOR
Ø	$ØØ	@	Inverse	Black
17	$11	Q	Inverse	Magenta
34	$22	"	Inverse	Dark Blue
51	$33	3	Inverse	Purple
68	$44	D	Flash	Dark Green
85	$55	U	Flash	Grey 1
1Ø3	$66	'	Flash	Medium Blue
119	$77	7	Flash	Light Blue
136	$88	H	Control	Brown
153	$99	Y	Control	Orange
17Ø	$AA	*	Normal	Grey 2
184	$BB	8	Normal	Pink
2Ø4	$CC	L	Normal	Light Green
221	$DD	]	Normal	Yello
238	$EE	.	L. Case	Aquamarine
255	$FF	?	L. Case	White

High Resolution

Low resolution allows you to control the colors of a 40 by 47 block array. A single lo-res block is equivalent to 56 hi-res pixels. High resolution (hi-res) lets you control an array of 280 by 192 pixels, resulting in an obvious improvement in detail. That's the good news.

Now for the bad news. There are 16 lo-res colors, but only eight hi-res colors (that includes two whites and two blacks).

HI-RES COLORS

Value	Description
0	black 1
1	green
2	blue
3	white 1
4	black 2
5	purple
6	orange
7	white 2

There are several problems with hi-res color, which are caused by both the way color is produced on the set (TV or monitor) and the way the hi-res screen buffer is formatted in memory. (See "Color".)

HGR initiates hi-res graphics by displaying hi-res page one, clearing it to black, and setting the mixed screen mode. HGR does this by activating a series of soft switches (addresses in memory that "switch" whenever they are POKEd or PEEKed). The screen soft switches let you switch between:

A. Graphics and text.
B. Full screen and mixed screen.
C. Page one and two.
D. Lo-res and hi-res.

(See table of screen soft switches.)

PEEKing or POKEing the following addresses will duplicate HGR soft switching: 49300, 49235, 49239, 49232.

To make certain that hi-res page one is used as well as displayed, POKE 230,32. To clear the screen to black, CALL 62450.

HGR2 initiates hi-res graphics on page two. It can be duplicated by PEEKing or POKEing these addresses: 49237, 49234, 49239, 49232. Be sure to POKE 230,64 so that drawing will also be done on page two. To clear it, CALL 62450.

Once graphics is initiated, it's time to draw.

Color is specified by HCOLOR = n, where "n" is any number from zero to seven (see hi-res colors above). Setting HCOLOR = to a number less than zero or greater than seven results in an error.

To plot on the hi-res screen, use HPLOT. It can be used in various ways:

HPLOT X,Y can draw a dot on the screen (if the HCOLOR is correct for that dot position).

HPLOT TO X,Y can put a line beginning with the last point HPLOTed and ending at the pixel whose position is X,Y.

HPLOT X1,Y1 TO X2,Y2 can put a line on the screen beginning on the pixel at the position X1,Y1 and ending on the pixel at X2,Y2. (Again, whether a line is actually drawn may depend on the HCOLOR specified. For more information, see "Color").

HPLOT X1,Y1 TO X2,Y2 TO X3,Y3 TO . . . Xn,Yn will plot one line after another, each beginning where the last line ended.

Because the screen positions are limited to 280 pixels across and 192 down, all values of X less than zero and greater than 279 are met with an error, and all Y values less than zero and greater than 191 are similarly met.

Other ways to place graphics on the hi-res screen include:

A. Poking values into the memory used by the hi-res page buffers.
B. Using shape tables (vector graphics).

Graphics placed on the hi-res page can be saved as a binary file. To save page one on disk, type

BSAVE name, A$2000, L$2000

or in decimal

BSAVE name, A8192, L8192

By changing the address (A) to $4000, you can save the picture on page two. There are also other hi-res "pages" that, though they cannot be directly displayed, can be saved.

Page Three	**A$6000**
Page Four	**A$8000**
Page Five	**A$10000**

DOS normally occupies hi-res page five.

Using Applesoft Hi-Res Routines from Machine Language

For those who enjoy working in assembly language, here are all the hi-res commands available from BASIC, as well as four additional ones. This section is geared for the advanced user who is already familiar with assembly language.

Zero Page Locations

First here are some zero page locations used.

$1A,1B Shape pointer used by DRAW and XDRAW.
$1C Last color used (HCOLOR converted to its color byte. See Color Byte Table).
$26,27 Address of byte containing X,Y point.
$30 The bit mask for the bit in the current byte.
$E0,E1 X-coordinate (0-279) in hex (low, high).
$E2 Y-coordinate (0-191) in hex.
$E4 Color being used (converted, see Color Byte Chart).
$E6 Current hi-res page being used ($20: page one, $40: page two).
$E7 Current SCALE (0-256).
$E8,E9 Location of shape table (low, high).
$EA Collision counter (used by XDRAW and DRAW).

Black and Blue

Here's an example of how to use some of the routines from assembly langauge.

```
JSR HGR          INITIALIZE THE SCREEN
LDX #$2          SET THE COLOR TO BLUE
JSR SETHCOL
JSR BKGND        MAKE THE ENTIRE SCREEN
                 BLUE
LDX #$0          USE BLACK TO DRAW LINES
JSR SETHCOL
LDA #$0          PLOT THE FIRST POINT AT 0,0
LDX #$0
LDY #$0
JSR HPLOT
LDX #$00         DRAW A LINE FROM LAST
                 POINT TO 50,128
LDA #$32
LDY #$80         Y-COORDINATE
JSR HLIN
RTS              EXIT TO CALLER
```

Remember, to use DRAW and XDRAW, point (X,Y) to the actual shape, not to the beginning of the shape table. This means that all calculations must be done by the user, to index into the shape table.

Basic Hi-Res Commands

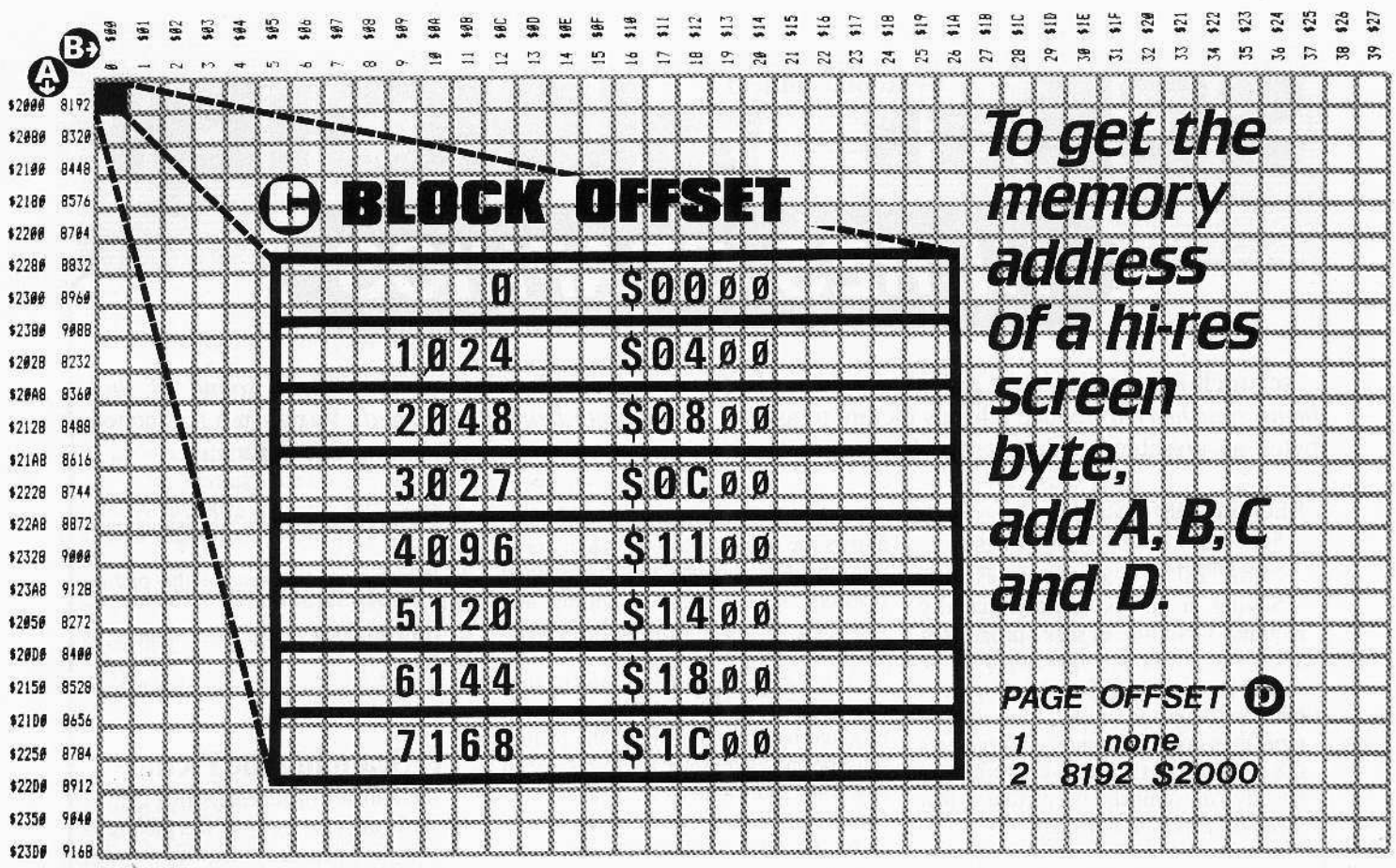
The following are the routines available for hi-res graphics.

HGR **$F3E2** Initialize and clear hi-res page one.
HGR2 **$F3D8** Initialize and clear hi-res page two.
HPLOT **$F457** Positions the cursor and plots a point. Enter with the Accum (A) = Y coordinate and the X register and Y register containing the X coordinate (low, high).
HLIN **$F53A** Draws a line from the last plotted or positioned point to (A,X) = X coordinate (low, high), (Y) = Y coordinate.
DRAW **$F601** Draws a shape. Enter with (X,Y) pointer to the actual shape to be drawn, not to the shape table itself. The accumulator should contain the ROTation factor. Uses current color and scale.
XDRAW **$F65D** Performs the XDRAW command, same entry as DRAW.
SETHCOL **$F6EC** Performs the HCOLOR command where the X register contains a color 0-7.
SCALE **$E7** Simply place scale factor here (STA).
ROT See the DRAW command.

Additional Commands

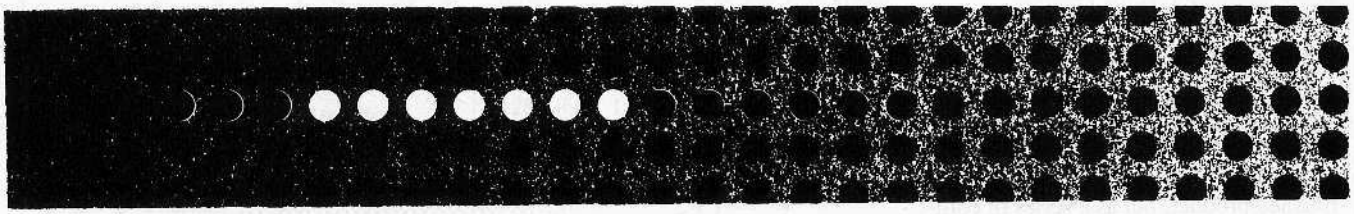
The following commands are not available from BASIC.

HPOSN **$F411** Moves the hi-res cursor without plotting a point. Enter same as HPLOT.
HFIND **$F5CB** Converts the current hi-res cursor's position to X and Y coordinates. Can be used to find where you are left after drawing a shape. After calling this routine, $E0,E1 is the X position (low,high) and $E2 is the Y position.
HCLR **$F3F2** Clears the current hi-res page to black.
BKGND **$F3F6** Clears the current hi-res screen to the last color plotted.

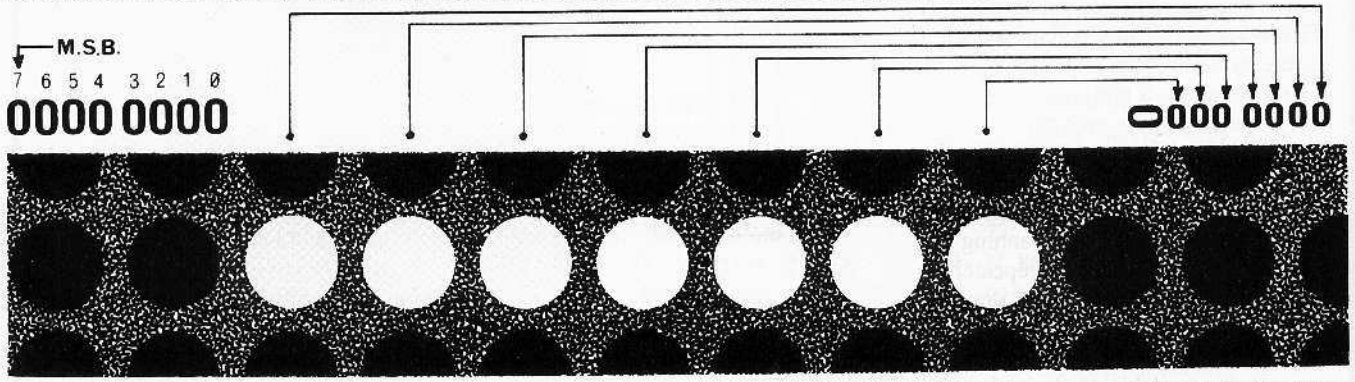


HI-RES SCREEN FORMATTING

1. Horizontally, the hi-res screen consists of only 40 bytes. In this illustration, one byte is illuminated and enlarged.



2. As can be seen, only seven bits are "lit". The eighth bit (also known as the MSB, Most Significan Bit) is the Color Bit. Although not shown, its value (zero or one) determines the color combination. For more information, see "Color".



3. When displayed on screen, the "dots" correspond to actual bit positions, but the positions are reversed. The last dot of the displayed byte is actually the first bit of the byte in memory. To be technically correct, the bit numbers are from zero to seven, rather than from one to eight. That means that the MSB (eighth bit) is bit seven.

SCREEN CRUNCHER

by Robb Canfield

HI-RES GRAPHICS

scrunch /skur-runch/ (*the sound of a hi-res bug being squished*) To reduce a hi-res picture to as few bytes as possible so that it can be unscrunched.

unscrunch /un-skur-runch/ (*the sound of that same bug being unsquished*) To return a scrunched picture to its original pattern. See Scrunch.

REQUIREMENTS:
48K Apple II (or Franklin Ace)
One Disk Drive

Saving a hi-res picture usually requires 34 sectors of disk space. That allows only about 14 pictures to be saved to the disk. Scruncher 1.0, a machine-language utility, can usually more than double the number of pictures (28 to 40, in fact!) per disk. It will also quickly "un-scrunch" the picture so that it can be displayed normally.

How To Scrunch

There are two distinctly different techniques used to reduce the amount of space required to store a picture.

One method saves only the commands used to draw the picture. An example would be: draw a circle at 90,90 with a radius of 20 and color it in with green. The picture (a green circle on the hi-res screen) is not saved as a finished product, but as a series of commands telling another program how to draw a picture. This method, used in many hi-res adventure games such as Wizard and the Princess, can reduce a picture by up to 90%.

Unfortunately, it requires:

1. a special editing program
2. drawing the picture in the fewest number of commands.

Another way is to condense the completed picture. It involves scanning the entire picture to look for "repeaters". This allows you to shrink any drawing you have made previously, and to use any one of the excellent drawing programs now available to draw the picture. It is possible to achieve savings of 30% to 90%, with an average of 46%.

I find this method more suitable as it allows me to use free-form when drawing a picture, whereas the first method restricts me from freely editing and changing my finished picture without an excessive amount of effort.

To actually reduce the amount of storage space (on disk or in memory) required by a hi-res picture, the picture must be encoded. Since the hi-res screen is nothing more than an array of 40 bytes across and 192 bytes down, the best way is to encode all the values that are repeated . . .

For example, in illustration 1 the values $00 and $3F are both consecutively repeating bytes:

$00 occurs twice in a row.
$3F occurs 4 times in a row.

Only $3F is a true repeater.

To encode a picture, 3 decisions must be made:

1. Which way to examine the picture data.
2. How to code in the least number of bytes.
3. What value to use as a marker byte.

1. Scanning the Picture

The number of repeaters that can be found is affected by how the picture is examined. There are at least three ways to look for repetitious values:

A. Sequentially through memory.
B. Horizontally through the picture (as it appears on the screen).
C. Vertically through the picture.

There is a difference between how the screen appears and how it's formatted in

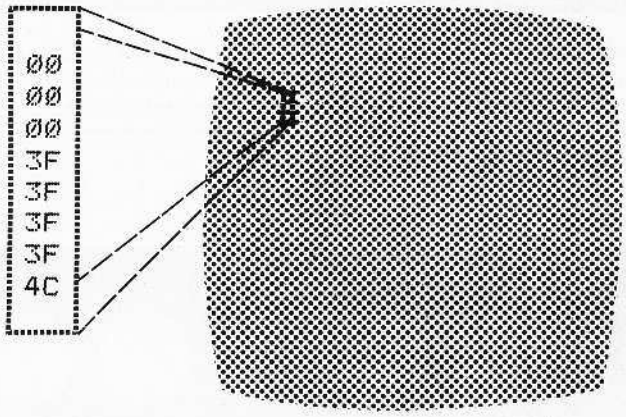


Illustration 1

Take a small part of the hi-res screen and look for values that repeat themselves consecutively. "Repeaters" are values (bytes) that are consecutively repeated 4 or more times. Other repeating bytes are not called repeaters.

memory.

To scan consecutively through memory would be inefficient because the hi-res screen is not oriented as consecutive bytes in memory. And since a hi-res picture is usually not a set of random values, no pattern on the screen would be easily coded unless it was examined in the order it appeared on the screen, not as it appears in memory.

The hi-res screen is only 40 bytes wide, so any value can be repeated horizontally only 40 times.

However, it's 192 bytes tall. Therefore, a value can be repeated up to 192 times. The vertical scanning method obviously provides a greater chance of finding a larger number of repeaters.

2. Coding the Repeaters

Now that a search method has been selected, it's time to create a coding method that uses the least number of bytes.

Scruncher uses three bytes to encode repeaters: a Code Marker, a Counter, and the Repeater.

The Code Marker informs the Unscruncher that encoded information is coming, much like the address marks used by DOS let it know data is coming. The second byte is a Counter that tells how many times to repeat the third byte, which is the actual repeat value that is encoded.

Illustration 2

FE 34 22

FE — the Marker Byte.
34 — Number of times to repeat.
22 — Byte to repeat.

In illustration 2, the three bytes are shown in their relationship to other unencoded bytes.

Because encoding takes three bytes, a repeater must be repeated at least 4 times consecutively. No space is saved when the repetition is less than four, and for every repetition greater than three, another byte of space is saved. If a value is repeated down the height of the picture, 189 bytes are saved (192 - 3 = 189).

3. Selecting the Marker

If any byte's value can be a part of the hi-res picture, what value can be used to indicate a coded sequence?

There is only one criterion used to select the best possible marker value: the number of times it is found in the hi-res picture.

The fewer times the byte value is found, the better that value will work. The reason for this is that every time a value is found which is the same as the repeat marker, it must be encoded, even

DEMO

```
10  NORMAL : TEXT : HOME
20 D$ =  CHR$ (4)
30  REM

RELOCATE?
40  IF  PEEK (103) = 1 AND  PEEK
    (104) = 96 THEN 60
50  POKE 103,1: POKE 104,96: POKE
    24576,0: PRINT D$"RUN DEMO"
60  NORMAL : TEXT : HOME : POKE 2
    30,32: POKE  - 16304,0: POKE
     - 16300,0: POKE  - 16297,0:
     POKE  - 16301,0
70  IF  PEEK (8 * 256 + 3) = 169 AND
     PEEK (8 * 256 + 4) = 255 THEN
    110
80  VTAB 22: PRINT "PLEASE WAIT W
    HILE I LOAD THE FILES"
90  PRINT D$"BLOAD PACK,A$803": PRINT
    D$"BLOAD UN-PACK,A$300"
100  REM
WHERE END OF COMPRESSED
    PICTURE IS
110 LO = 8 * 256 + 15 * 16 + 12
120  HOME : POKE  - 16304,0: VTAB
    22: PRINT "COMPRESS/DECOMPRE
    SS (C/D)?";: GET A$: PRINT
130  IF A$ <  > "C" AND A$ <  > "
    D" THEN  PRINT "ILLEGAL ENTR
    Y": GOTO 120
140  IF A$ = "D" THEN 360
150  REM

COMPRESS OPTION
160  VTAB 24: PRINT "ENTER NAME O
    F HI-RES PICTURE TO COMPRESS
    "
170  HTAB 5: INPUT "> ";NA$
180  IF  LEFT$ (NA$,1) = D$ THEN
     TEXT : HOME : PRINT NA$: GET
    A$: HOME : POKE  - 16304,0: GOTO
    160
190  IF NA$ = "" THEN 220
200  PRINT D$"BLOAD"NA$",A$2000"
210  REM
COMPRESS PICTURE
220  CALL 8 * 256 + 3
230 LE =  PEEK (LO) +  PEEK (LO +
    1) * 256 - 16384
240  PRINT "LENGTH OF COMPRESSED
    PICTURE:"LE
250  PRINT "NUMBER OF BYTES SAVED
    :"8192 - LE
260  PRINT "PERCENTAGE DIFFERENCE
    "100 -  INT (LE / 8192 * 100
    )"%"
270  PRINT "BSAVE THIS COMPRESSIO
    N (Y/N)? ";: GET A$: PRINT
280  IF A$ <  > "Y" THEN  HOME : GOTO
    120
290  PRINT "UNDER WHAT NAME ('.C'
     IS APPENDED)"
300  HTAB 5: INPUT "> ";NA$
310  IF  LEFT$ (NA$,1) = D$ THEN
     TEXT : HOME : PRINT NA$: GET
    A$: POKE  - 16304,0: HOME : VTAB
    22: GOTO 290
320  IF NA$ = "" THEN  HOME : GOTO
    120
330  PRINT D$"BSAVE"NA$".C,A$4000
    ,L"LE
340  GOTO 120
350  REM

DECOMPRESS OPTION
360  PRINT "COMPRESSED PICTURE ('
    .C' IS APPENDED)"
370  HTAB 5: INPUT "> ";NA$
380  IF  LEFT$ (NA$,1) = D$ THEN
     TEXT : HOME : PRINT NA$: GET
    A$: POKE  - 16304,0: HOME : VTAB
    22: GOTO 360
390  IF NA$ = "" THEN 410
400  PRINT D$"BLOAD"NA$".C,A$4000
    "
410  CALL 3 * 256: REM  UNPACK PI
    CTURE
420  GOTO 120
```

if it is found only once (increasing the code instead of decreasing it!).

If this value was not encoded, there would be no way to tell the difference between a marker and a byte with the value of the marker, since they both are the same.

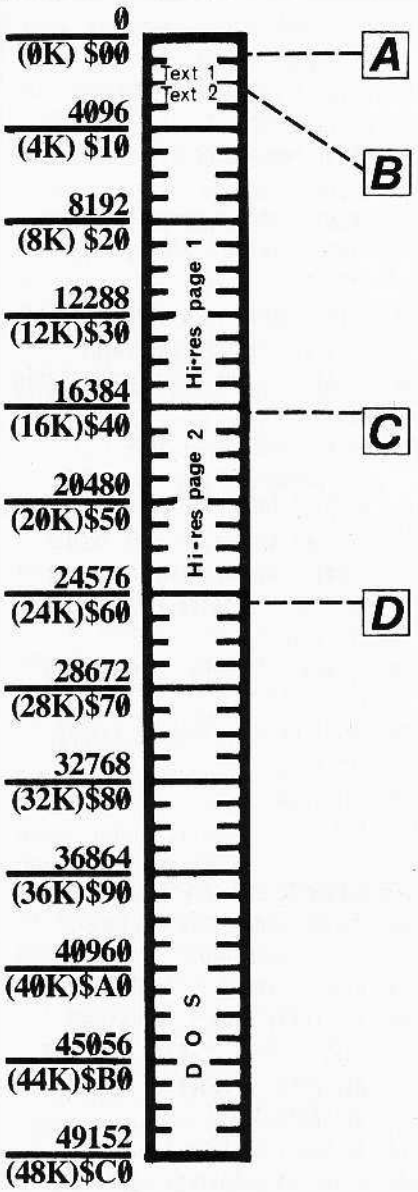
Illustration 3

FE 01 FE

FE — the Marker Byte.
01 — Number of times to repeat the marker
FE — the Marker Byte.

Illustration 3 shows this. Imagine the illustration as a section of the encoded picture, and $FE as the selected marker byte. The decoding program would find the $FE, and assuming it to be a marker, would repeat the value $22, 54 ($34) times. The program has no way of knowing that this byte was not a marker for a repeater. So to encode the information properly it becomes necessary to encode all values found that are the same as the marker byte.

To solve the problem of selecting the best marker value, Scruncher searches the entire hi-res picture looking for the value that is found the least number of times. This value is then used as the marker byte and is stored as the first value in the code buffer. Unscruncher then looks at this first value and uses it as the code marker, allowing each picture to have the best possible value as its code marker.



C Code Buffer

D DEMO

How Scruncher Works

Scruncher examines the hi-res picture vertically just as it appears on the screen. It checks each byte to see if it is the value of the marker or if it has been encountered four times in a row. If either condition is true, then coding takes place and the code is moved to the code buffer. If both conditions are false, then it stores that unaltered value in the code buffer. This process continues until all 40 columns are transferred.

If no value is repeated more than three times, there are still 512 bytes (two sectors or one-half of a kilobyte) saved because the program ignores the presence of "hidden" bytes on the hi-res screen.

But nearly all pictures have values that occur more than three times. Generally, a saving of 20% or better occurs.

Hex Dump for UN-PACK **A**

```
0300- A2 00 A0 40 8C 30 03 A0
0308- 00 8C 2F 03 20 2E 03 85
0310- FE 20 2E 03 C5 FE F0 05
0318- 20 3A 03 90 F4 20 2E 03
0320- 85 00 20 2E 03 20 3A 03
0328- C6 00 D0 F9 F0 E3 AD FF
0330- FF EE 2F 03 D0 03 EE 30
0338- 03 60 48 98 48 29 C0 8D
0340- 68 03 4A 4A 0D 68 03 8D
0348- 68 03 68 8D 69 03 0A 0A
0350- 0A 2E 69 03 0A 2E 69 03
0358- 0A 6E 68 03 AD 69 03 29
0360- 1F 09 20 8D 69 03 68 9D
0368- FF FF C8 C0 C0 90 07 A0
0370- 00 E8 E0 28 B0 01 60 68
0378- 68 60
```

Hex Dump for PACK **B**

```
0803- A9 FF 8D 09 09
0808- A9 00 8D 07 09 85 FE A9
0810- 00 8D 08 09 8D 1D 08 A9
0818- 20 8D 1E 08 AD FF FF CD
0820- 07 09 D0 05 EE 08 09 F0
0828- 29 EE 1D 08 D0 EE EE 1E
0830- 08 AD 1E 08 C9 40 D0 E4
0838- AD 08 09 D0 08 AD 07 09
0840- 85 FE 4C 59 08 CD 09 09
0848- B0 08 8D 09 09 AD 07 09
0850- 85 FE CE 07 09 D0 B8 A5
0858- FE 8D 00 40 A2 00 86 00
0860- 86 01 A0 40 8C FD 08 A0
0868- 00 8C A2 08 A9 01 8D FC
0870- 08 A9 20 8D A3 08 98 48
0878- 29 C0 8D A2 08 4A 4A 0D
0880- A2 08 8D A2 08 68 8D A3
0888- 08 0A 0A 0A 2E A3 08 0A
0890- 2E A3 08 0A 6E A2 08 AD
0898- A3 08 29 1F 09 20 8D A3
08A0- 08 BD FF FF 24 01 30 08
08A8- 85 03 A9 00 85 01 30 07
08B0- C5 03 F0 03 20 CC 08 E6
08B8- 00 C8 C0 C0 90 B8 20 CC
08C0- 08 E8 E0 28 F0 26 A0 00
08C8- 84 01 F0 AA 48 84 04 A4
08D0- 00 C0 04 90 18 A5 FE 20
08D8- FB 08 98 20 FB 08 A5 03
08E0- 20 FB 08 A9 00 85 00 68
08E8- A4 04 85 03 60 A5 03 C5
08F0- FE F0 E2 20 FB 08 88 D0
08F8- FA F0 E8 8D FF FF EE FC
0900- 08 D0 03 EE FD 08 60
```

Entering the Scruncher

There are three sections in the Scruncher program. Two are machine language programs (PACK and UN-PACK), and the last one is in BASIC. The demo allows you to load a picture into memory and scrunch (using PACK) or un-scrunch it (using UN-PACK).

Directions for Entering PACK

1) Enter the monitor.
CALL-151
2) Type in the hex-dump for PACK.
3) Save PACK to the disk. Do not return to BASIC. BSAVE can be done from the monitor. Just . . .
BSAVE PACK,A$803,L$106

Directions for Entering UN-PACK

1) Type in the hex-dump for UN-PACK.
2) Save UN-PACK to the disk. (You can do this from the monitor or from BASIC.)
BSAVE UN-PACK,A$300,L$
3) Return to BASIC. (If you haven't done so already.)
3D0G

Directions for Entering DEMO

1) Reset Applesoft pointers.
FP
2) Type the DEMO Applesoft listing.
3) Save the program.
SAVE DEMO

To use the program, simply RUN DEMO. It will first relocate itself (more on that later), and then load the programs PACK and UN-PACK. You will notice that the hi-res screen is now displayed.

At this time you will be asked either to "Compress" or "Decompress" a picture. The "Compress" option will PACK (encode) the picture so that it takes up less room. "Decompress" will UN-PACK (decode) a compressed picture.

Type either C or D.

If you decide to compress a picture, you will be prompted to enter the picture's name.

If you simply press RETURN, the current picture (as shown on the hi-res page) will be compressed.

continued on page 61

PACK

```
1000 *-------------------------------
1010 * HI-RES PICTURE PACKER PROGRAM
1020 *
1030 *             BY
1040 *
1050 *          ROBB CANFIELD
1060 *
1070 *          NOV. 15 1982
1080 *
1090 *-------------------------------
1100
1110
1120
1130 COUNTER   .EQ $00       NUMBER OF TIMES TO REPEAT
1140 FIRST.TIME.RAN .EQ $01 ($00 MEANS FIRST RUN)
1150 TABLE     .EQ $3        LOCATION OF BYTE TO REPEAT
1160 YSAVE     .EQ $4        Y-REG SAVE AREA
1170
1180 REP.CHAR .EQ $FE        MARKER CHARACTER
1190
1200
1210           .OR $803
1220           .TF PACK
1230
1250
1260
1270 *-------------------------------
1280 * FIND BEST REPEAT BYTE, BY
1290 * SEARCHING THRU THE HIRES SCREEN
1300 *-------------------------------
1310
1320
1330 SEARCH
1340           LDA #$FF      RESET LAST.COUNT
1350           STA LAST.COUNT
1360           LDA #$00      RESET SEARCH POINTERS
1370           STA CURRENT
1380           STA REP.CHAR
1390 .0
1400           LDA #$00
1410           STA CURRENT.COUNT
1420           STA .1+1
1430           LDA #$20
1440           STA .1+2
1450
1460 .1        LDA $FFFF     GET A BYTE FROM HIRES SCREEN
1470           CMP CURRENT   SAME AS REPEAT BYTE?
1480           BNE .2        NO SO CONTINUE
1490           INC CURRENT.COUNT
1500           BEQ .3
1510 .2        INC .1+1      INCREMENT ADDRESS
1520           BNE .1
1530           INC .1+2
1540           LDA .1+2      IS IT 40?
1550           CMP #$40
1560           BNE .1
1570           LDA CURRENT.COUNT GET BEST REPEATE VALUE
1580           BNE .5
1590           LDA CURRENT   GET BYTE
1600           STA REP.CHAR
1610           JMP .4
1620 .5        CMP LAST.COUNT
1630           BGE .3
1640           STA LAST.COUNT
1650           LDA CURRENT   SAVE REPEAT BYTE (NEW ONE)
1660           STA REP.CHAR
1670
1680 .3        DEC CURRENT   GET NEXT BYTE TO CHECK
1690           BNE .0
1700           LDA REP.CHAR SAVE REPEAT BYTE IN BUFFER
1710 .4        STA $4000
1720
1730
1740
1750 *-------------------------------
1760 * START TO COMPRESS PICTURE.
1770 *-------------------------------
1780
1790
1800
1810           LDX #$00      RESET HORIZONTAL OFFSET
1820           STX COUNTER   RESET REP COUNT
```

continued on page 61

THE UFO FACTORY

by Bev R. Haight

When designing a game, one must create game images that do not replicate (copy) those already closely identified with another game. That means that a game designer must not use Pac-Man, because that would be a copyright infringement. It would be sort of like using Mickey Mouse without Disney approval.

Besides, unless the game is meant to satirize another game's images, it is very unprofessional to copy images created by another designer.

Image duplication is easy to avoid in the case of large shapes, but it is almost impossible to create an original design for small shapes (especially when working with as small a shape as seven by eight pixels).

The original ***Night Falls*** contained saucer-style entities that had been overused as an invader image. Of course, there were only a few possible designs for saucer-like entities on the small scale that the game required. These early saucers were simple (both in code and in plotting) in order to maintain the game's speed. They are shown in the Night Falls advertisement. The early saucers are no longer used in the version of the game being sold. In fact, no saucer shape is stored in the program because Night Falls creates its own saucer-shapes as they are required. The game comes with its own UFO Factory.

The program accompanying this article is similar to the Night Falls UFO Factory only in principle. In Night Falls, the UFOs come in a variety of color combinations and shapes. This version creates UFOs in white only. Although limited to a single color, the UFO Factory will still create hundreds (if not thousands) of UFOs with a very small likelihood of duplicating itself, much less another game's images.

The size (width and height) of the UFOs depends upon a number (one through six) that the user selects. The Factory chooses the number of UFOs to display (in rows and columns) and then randomly creates them. After the UFOs have been created, you can quit or try another size. Just follow the prompts.

Those who have been around since the first Hardcore Computing will note that the UFOs resemble the old "Ink Blot" shapes, but on a much smaller scale.

The basic algorithm is the same, too. For each vertical line needed to draw the UFO, the Factory selects two random numbers, from zero to the selected width. After these two points have been moved to the left and right of the center of the figure, a line is drawn between them. There is a calculated overlap added to one and subtracted from the other. And finally, after drawing all the line pairs, an unpaired line is randomly placed in the figure to make sure that the two sides are connected.

In Night Falls, these numbers are stored in an array so that the UFOs can be animated, created, and destroyed. In the Factory, they are only calculated and plotted.

What can be done with the UFO Factory? An addition used to change the Factory into a game would be interesting, letting you take the model for a test drive through the stars.

But basically, the Factory is presented here to show how design problems can be aided by the computer. Other Factory algorithms were tested (incorporating such parameters as 'rotate-ability' and scale, and the usual assortment of un-saucer-like shapes, such as globular 'bugs' and eye-balls), then discarded. The easiest shapes, of course, were bilaterally symmetrical (left and right mirror images). You could also try shapes that are radially symmetrical.

In fact, whenever innumerable variations on a particular theme are sought, use your computer to generate them. There is no need to use random parameters, even though they come in handy with a shape as common as a UFO.

If you need game shapes and fear that you may unknowingly copy another game's images, just create your own shape factory and discover how many shapes there really are out there, unused, free for the making . . .

UFO Generator

```
0  REM
UFO GENERATOR

1  REM
ADAPTED FROM
2  REM
"NIGHTFALLS"
10  HCOLOR= 7: GOTO 1000
49  REM
PRINT ROUTINE

50  VTAB 21: CALL  - 868: HTAB 10
    : RETURN
99  REM
DRAW UFO

100 U1 =  RND (1) * WW + (WW / 3)
110 U2 =  RND (1) * WW - (WW / 3)
120  HPLOT X + U1,Y TO X + U2,Y
130  HPLOT X - U1,Y TO X - U2,Y
190  RETURN
199  REM
LOGO

200  TEXT : HOME
210  INVERSE : VTAB 23: PRINT  SPC(
     39)
250  VTAB 23: HTAB 10
260  PRINT "THE U.F.O. FACTORY"
270  VTAB 24: HTAB 2: NORMAL
280  PRINT "ALL THE MODELS IN THE
      WHOLE GALAXY!";
290  NORMAL : RETURN
999  REM
ENTRANCE
1000  GOSUB 200
1100  GOSUB 50
1110  PRINT "MODEL OF UFO? (1-5)"
     ;
1120  GET A$: IF A$ = "X" THEN 20
     00
1130 WW =  VAL (A$) + 3: IF WW <
     4 OR WW > 9 THEN 1120
1150  GOSUB 50: HTAB 5
1160  PRINT "THIS IS THE ";
1170  ON WW - 3 GOSUB 2100,2200,2
     300,2400,2500,2600
1199  REM
BEGIN DRAW CYCLE

1200  HGR
1250 ZZ = 250 / (WW + 4) / 2
1300  FOR W = 1 TO ZZ
1350 X = W * (WW + 4) * 2
1400  FOR H = 1 TO 150 / (WW + 4)
1500  FOR N = 1 TO WW - 1
1510 Y = H * (WW + 4) + N
1520  GOSUB 100
1540  NEXT
1610 Y = Y -  RND (1) * WW + 2
1620 U1 = WW:U2 =  - WW
1630  HPLOT X + WW,Y TO X - 1 - W
     W,Y
1650  NEXT : NEXT
1799  REM
AGAIN?

1800  GOSUB 50
1810  PRINT "PRESS KEY FOR NEW UF
     O";
1850  GET A$
1860  IF A$ = "X" THEN 2000
1890  GOTO 1000
1900  END
1999  REM
EXIT

2000  GOSUB 50
2010  PRINT "THANK-YOU FOR VISITI
     NG"
2020  GOSUB 210
2090  VTAB 1: END
2099  REM
UFO TYPE

2100  PRINT "(1) COMPACT IMPORT":
      RETURN
2200  PRINT "(2) DOMESTIC COMPACT
     ": RETURN
2300  PRINT "(3) GALACTIC RUN-ABO
     UT": RETURN
2400  PRINT "(4) FAMILY-SIZE UFO"
     : RETURN
2500  PRINT "(5) LUXURY LIMOSINE"
     : RETURN
2600  PRINT "(6) GAS-HOG": RETURN
```

(NOT Copy-protected!)

Going from 16 lo-res colors to only six hi-res colors (if you consider both types of black as black, and both whites as white) is sometimes difficult to accept. So it might come as a disappointment to also discover that the so-called "higher resolution" of 280 dots across by 192 high must be cut in half whenever color becomes an important element in the hi-res display.

In otherwords, there are only 140 horizontal points, though there are still 192 points vertically. And you are limited as to what colors can occupy certain adjacent horizontal positions. Why is there so much trouble with the horizontal points of color?

The answer lies in the way the Apple hardware uses the color display (see "Why Only Blue, Green, Purple and Orange?") and the way the Apple memory is displayed.

The manner in which Apple memory is shown on the screen is the key to understanding how best to use the available color for hi-res displays.

Unlike the text and lo-res displays, when the hi-res buffer is put onto the screen, it is a more "direct" process (called "bit mapping"). In text, a single byte of memory becomes a single character on the screen. One way to describe the text display, is to describe it as 40 bytes wide by 24 bytes high. In lo-res, a single byte of memory is split into two nybbles and displayed as two stacked blocks of color. Again, it is only 40 bytes across and 24 high, but the resolution is doubled because vertically there are twice as many controllable blocks.

In contrast, the hi-res page is eight times as detailed in the vertical dimension: 8 times 24 is 192. But horizontally, it's still only 40 bytes across. And only seven of the eight bits that make up that byte are displayed. Horizontally, the hi-res image is seven times as detailed as the lo-res image. But because of the 40-byte horizontal capacity, certain color limitations are created.

The "pure" hi-res colors are limited to two blacks, two whites, a blue, a green, a violet and a red. But certain horizontal positions are limited to certain color combinations. Starting at the left side of the screen, the first hi-res dot is at position zero, the next is one, then two and so on to the right until the right margin is reached at point number 279. All odd-numbered points can be either green or red (or black), while all even-numbered points can be blue or violet (or black). White is created by any two horizontally adjacent dots.

Color by the Byte

	EVEN	ODD	TOGETHER	ABSENT
SET	Blue	Red	White 1	Black 1
NOT SET	Purple	Green	White 2	Black 2

Whether that even-numbered dot is blue or violet depends upon the invisible eighth bit of that byte of memory. The red or green is similarly controlled.

That invisible bit is called the High Bit (MSB, Most Significant Bit). When it is set (equal to one), the even-numbered points are blue and the odd are red. When it is not set (a zero value), the even points are violet and the odd are green (see "Color by the Byte").

That simple pattern would be easy if it were not for that same invisible eighth bit . . . and with only seven bits displayed, the pattern for odd/even is just the opposite for bytes sitting side-by-side. It takes two horizontally adjacent bytes to create a repeating pattern: 1010101 0101010. If the first of the 40 bytes across is zero, and the last (to the far right) is 39, then even bytes have all blue/violet first, third, fifth, and seventh bits, and odd bytes have all red/green for those same bits! If that isn't confusing, you should be writing this introduction to color.

Because the two "complementary" color groups (high bit set and high bit not set) are mutually exclusive in any particular byte, green and red cannot be put in the same byte (see Impossible Color Chart). When the high-bit-set color is placed in the same byte as a high-bit-not-set color (even black!), an interesting color change occurs because the high bit changes.

Impossible Color Combinations
(in the same byte)

first color	the second color can be any of these:	the first color will change from:	to:
green	(red, blue, white 2, black 2)	green	red
violet	(red, blue, white 2, black 2)	violet	blue
red	(green, violet, white 1, black 1)	red	green
blue	(green, violet, white 1, black 1)	blue	violet

Now that you are fully exasperated at the Apple colors, take heart. It is possible to get more colors . . . some claim only 21 colors total, others claim up to 256.

"Twenty-One Colors" is a program that creates "artificial" colors by placing two colors next to each other . . . vertically.

```
0 REM 21 COLORS!
10 HGR
20 XX = 28:YY = 20
30 FOR X = 0 TO 7
```

```
40 FOR Y = 0 TO 7
50 FOR Z = 0 TO YY STEP 2
60 HCOLOR= X: HPLOT X * XX,Y * YY + Z TO X * XX + XX
   ,Y * YY + Z
70 HCOLOR= Y: HPLOT X * XX,Y * YY + Z + 1 TO X * XX +
   XX,Y * YY + Z + 1
80 NEXT : NEXT : NEXT
```

In a way, there are 256 colors available, although only those that can be specified by HCOLOR= are solid colors. In order to get the other "colors", use these locations:

Hex	Decimal	Use
$1C	28	Color byte used to alter background color (CALL 62454).
$E4	228	Color byte used by HPLOT and DRAW.

To use these locations, just POKE in a value from zero to 255.

To demonstrate the other "colors," try the programs entitled "256 Background Colors" and "256 HPLOT Colors."

```
10 REM  256 BACKGROUND COLORS
20 HGR: HOME
40 FOR A = 0 TO 256
50 POKE 28, A
60 CALL 62454
90 VTAB 24: HTAB 15
100 NEXT : END
```

Did you notice that only certain colors are solid?

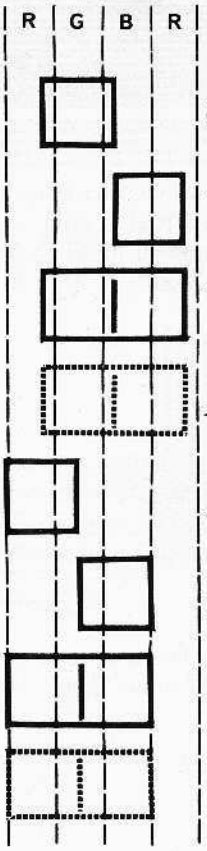
Color Byte Chart

Value to Poke	HCOLOR Value	Color Name
0	0	Black 1
42	1	Green
85	2	Blue
127	3	White 1
128	4	Black 2
170	5	Purple
213	6	Orange
255	7	White 2

To HPLOT with 256 colors, add these lines to the prior program:

```
10 REM 256 HPLOT COLORS
30 B = 128
50 POKE 228, A
60 IF A > B THEN 110
70 HPLOT 0, A TO B, A
110 HPLOT B, A - B
120 HPLOT TO 279, A - B
130 GOTO 100
```

? Why only...



GREEN 1/3 Red, 1 Green, 1/3 Blue

VIOLET 3/4 Blue, 3/4 Red

WHITE 1 1 Red, 1 Green, 1 Blue

BLACK 1

RED 1 Red, 1/2 Green

BLUE 1 Blue, 1/2 Green

WHITE 2 1 Red, 1 Green, 1 Blue

BLACK 2

When the hi-res colors are described as red, blue, green, purple, black and white, but you can't quite adjust the color set to show all those colors simultaneously... but don't worry. The red is really quite a bit orange, the blue is a bit greenish, the green (with its share of red and blue) is a light green, and the violet is quite pink! The reason why the colors are not "pure" is that there are only three so-called "pure" colors and the Apple does not display any of them by themselves.

All the Apple colors are really combinations of these basic colors: red, green, and blue. These are not really true red, green or blue but are rather the primary additive colors of light. The colors we normally associate with primary colors (yellow, blue and red) are the primary subtractive colors of pigments. If all the primary subtractive (pigment) colors are mixed, the result is black by subtraction. Mixing the primary additive colors result in white by addition.

In the illustration above and on your set, these additive colors are in vertical bands (though they will appear as dots, hexagrams, or bars when examined closely) that are labeled here as R,G,B (for Red, Blue, Green).

The Apple 'pixel' is actually not one, but one and a half screen pixels (colors). That means that the Apple colors are all color combinations. White is, of course, still composed of all three colors in equal proportion. And black is the absence of color.

When the Color Bit is set, the Apple pixels move three quarters of a color to the left, hence the color change. Green becomes red, violet becomes blue. White and black remain the same except that they are also shifted to the left.

VECTOR Graphics

Applesoft BASIC comes equipped with its own hi-res shape storage and display routines. Instead of HPLOTing shapes, programmers can use shape tables and all the simple commands that go with them:

DRAW
XDRAW
SCALE
ROT

A shape table is a collection of shape definitions, which in turn are a collection of drawing instructions called vectors (see ***Hardcore Computing*** Vol. 1, No. 3). These shapes can be placed anywhere on the hi-res screen by using the simple command:

DRAW n AT X,Y

For example, if a "Pac-Man" is stored in a shape table as shape definition number one, it can be drawn in the center of the screen by entering:

```
XDRAW 1 AT 140, 90
```

or . . .

```
HCOLOR = 3
DRAW 1 AT 140, 90
```

To erase it, you can:

```
XDRAW 1 AT 140, 90
```

or . . .

```
HCOLOR = 0
DRAW 1 AT 140, 90
```

Unlike HPLOT, no matter how complex or simple the shape, it can be easily drawn using the same command.

Other features not available to HPLOT, but part of the vector graphic subroutines are SCALE AND ROT.

To increase the size of the shape being displayed, you need only change the scale. For example, to make your Pac-Man into a giant . . .

```
SCALE = 10
XDRAW 1 AT 140, 90
```

SCALE can be set equal to zero through 255, with each increment from one to 255 being that much larger than the original shape. Setting SCALE to zero does not make it smaller. Instead, a scale of zero acts like 256! (To see SCALE in action, check out the program ***Design Plus,*** by Neil Taylor.)

ROT is the command for rotation. It allows the user to turn the shape around. To turn it 90 degrees to the right, just . . .

```
ROT = 16
XDRAW 1 AT 140, 90
```

Rotation can be used to spin your shape around its origin (the start of the shape's first vector). It can also be set equal to zero to 255, but whether the shape actually changes orientation depends upon its scale. At SCALE = 1, there are only four effective rotations.

Degree	ROT =
0	0 through 15
90	16 through 31
180	32 through 47
270	48 through 63
360 (0)	64 through 79
450 (90)	80 through 95
540 (180)	96 through 111
720 (270)	112 through 127

and so on . . . through 255

That means that at SCALE = 1, even if you set ROT = 15, the shape will still act as if you had set ROT = 0. It will change, though, at ROT = 16.

At SCALE = 2, these are the effective rotational values:

0, 45, 90, 135, 180, 225, 270, and 315.

At SCALE = 3 there are 16, at SCALE = 4 there are 32, and finally at SCALE = 5 and higher, all 64 rotations are available. In other words, at SCALE = 1 through SCALE = 5, the number of rotations doubles each time the value of SCALE increases by one.

Of course, ROT = 0 is the same as ROT = 64, which is the same as ROT = 128 and ROT = 192.

It is possible to store the entire character set in a very large shape table. But, unlike the normal characters on the text screen, the characters are alterable and they can be rotated and enlarged with a couple of simple commands. In this way, text can be set at various angles and sizes (subject to the limitations described above).

A special feature of shape tables is the XDRAW command. Unlike DRAW, XDRAW does not need to have an HCOLOR. It will draw a shape by exclusively ORing it; that is, by applying the shape on screen so that all parts placed over an "unlit" pixel will flip the pixel on, and all parts shown over a "lit" pixel will flip it off.

XDRAW lets you draw without erasing the background. It also lets you un-draw the same shape the same way, preserving all overlapping images.

Both DRAW and XDRAW using scale and rotation are used in ***NightFalls*** (***Hardcore Computing*** Vol. 1, No. 3).

VECTOR GRAPHICS

Shimmering Shapes

by Neil Taylor

Page-flipping is most often used to remove the "shimmering" effect caused by animation techniques that require undrawing and redrawing of objects in motion. By drawing on an undisplayed hi-res buffer, then displaying the page after all the undrawing and redrawing is finished, and then drawing on the other page (formerly displayed, but now "hidden"), exceptionally smooth animation can be achieved . . .

But page-flipping can be taken one step further than simply "smoothing" animation. By flipping between pages at a very high rate, a most unusual effect is created: the two pages seem to merge.

And speeding up this page-flipping even further creates a still more startling effect . . .

Design Plus includes a page-flipping routine that switches between the two hi-res pages at a rate of approximately forty-three thousand times per second. This produces an unusual shimmering effect on the monitor by taking advantage of the monitor's (or television's) inability to display a screen as quickly as the computer can. The computer changes the display before the monitor has shown all of the previous picture, causing only fragments of each screen to be viewed. The solid-looking parts of the picture are those areas which are the same on both screens.

The main purpose of Design Plus is to simply create interesting effects on your Apple. Most of the program is written in subroutines, including the drawing, incrementing, and checking of the keyboard. These could all be combined, but the separate subroutines make it easier to use page-flipping, and to change the program.

The flow of control within the program is easy to understand if you know the sequence used in page-flipping. Figure One shows the general procedure used in page-flipping and demonstrates the basic principles behind these procedures. The right-hand column shows the equivalent instructions from the program.

The next step is to know how to do this on the Apple. Page-flipping requires both the ability to draw on either page and the ability to display either page. The Apple can fill both of these requirements.

Figure Two contains the necessary locations for page-flipping on the apple, and an individual explanation of each.

The following variables are also in the program but are not unique to the page-flipping routine:

SC—SCALE of the square
INC—The increment that is added to the SCALE
A$,K%,A,B—Scrap variables

The shape table used has a center that remains stationary when the table is rotated. The shape table is poked into memory at hi-res page 4 and the pointers at $E8/232 are set so that you don't have to worry about them. For those who are curious, the table is printed here.

8000: 01 00 04 00 38 36 2D 24 07 00
E8:00 80 {the pointers}

The page-flipper routine is called through the step routine in the section which reads the keyboard. The machine language flipper is poked into memory directly following the shape table. It does not depend on

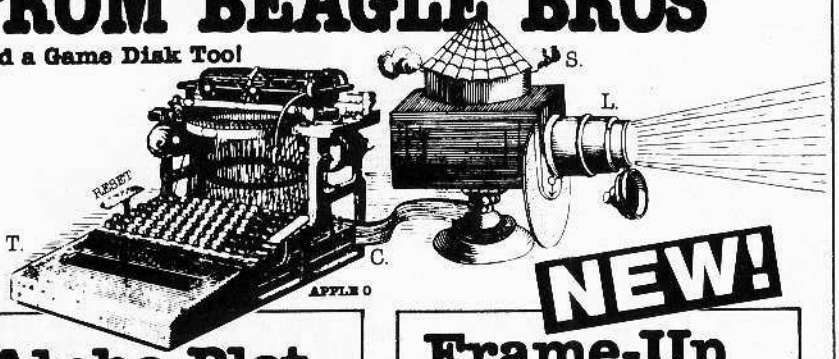
continued on page 34

Figure 1

I	**Clear both pages**	**HGR: HGR2**	
II	**Draw on page 1**	**POKE DRW,D1**	**GOSUB 190**
	A. Display page 1	**POKE S1,0**	
	B. Change the picture	**GOSUB 220**	
III	**Draw on page 2**	**POKE DRW,D2**	**GOSUB 190**
	A. Display page 2	**POKE S2, 0**	
	B. Change the picture	**GOSUB 220**	
IV	**Loop back to I**	**GOTO 20**	

NEW APPLE UTILITIES FROM BEAGLE BROS

And a Game Disk Too!



NEW!

Alpha Plot

Hi-Res Graphics/Text Utility
by Bert Kersey & Jack Cassidy

Here are a few of Alpha Plot's useful graphics features. Compare with other graphic utilities at any price—

HI-RES DRAWING: Create hi-res pictures and charts with text, on both pages; all **appendable to your programs**. Optional Xdraw cursor (see lines before drawing). Mix colors & Reverse (background opposite). Circles, Boxes, Ellipses; filled or outlined. **Compress Hi-Res to 1/3 disk space.** Relocate any portion of an image anywhere on either page. Superimpose too & convert hi-res to lo-res for colorful abstracts!

HI-RES TEXT: Beautiful upper/lower case with descenders (no hardware required). Color and reverse characters positionable anywhere (no vtab/htab limitations). Professional-looking **proportional spacing** and adjustable character height and letter spacing. Sideways typing for graphs too!

$39.50
- ☐ Unprotected disk (48K min.)
- ☐ Beagle Bros Apple Tip Book #4
- ☐ Peeks & Pokes Chart

NEW!

Beagle Bag!

12 Games on One Big Disk
by Bert Kersey

Twelve great games from our classic Beagle Bros collection—TextTrain, Slippery Digits, Wowzo, Magic Pack, Buzzword... Almost all of our "Game Pack" games have been updated and re-released on one jam-packed unprotected disk! **ALSO INCLUDED** is our "Beagle Menu" greeting program (description under "Typefaces" disk on this page).

Compare Beagle Bag with any 1-game locked-up disk on the market today!

All 12 games are a blast, the price is right, the instructions are crystal clear, AND the disk is copyable! You can even list the programs to see what makes them tick!

$29.50 Unprotected. Paddles NOT required. Beagle Menu works with all normal DOS disks. Includes Peeks/Pokes Chart.

Frame-Up

Graphics Display Utility
by Tom Weishaar

Frame-Up is a very-high-speed Apple "slide projector" utility that lets you create professional-looking displays of intermixed hi-res, lo-res and text pages **on any Apple.** Frame-Up is very easy-to-use and above-all **FAST,** allowing you to load hi-res pictures, for example, in **2½-seconds;** that's three-times faster than normal! Paddles or keyboard are used to change images in forward or reverse order, skipping pages if you want. OR presentations may be left unattended, with **each page individually timed** to appear and remain on the screen from 3 to 99 seconds, as you choose.

Frame-Up includes a sophisticated black and white **text screen editor** that lets you create text "slides" as part of your show. You can even add type "live" on the screen during your presentations. Up to 17 hi-res or 136 lo-res/text pages may be stored per disk. One or two drives are supported. The order and timing of your graphics and text images may be easily (and instantly!) arranged and rearranged. Frame-Up includes a **display module which may be copied** and distributed to your associates so they can run your display, as you designed it, on their Apple or ANY Apple!

Frame-Up is ideal for store displays, presentations to the boss, club programs, trade show booths, product demos, promotions, seminars, conventions, classes, and so on.

$29.50 Machine language. Unprotected. 48K minimum. Peek/Poke Chart included.

BEAGLE BROS DISKS ARE UNLOCKED AND UNPROTECTED. THIS MEANS EVERY PROGRAM IS INSPECTABLE, CUSTOMIZABLE, IF YOU WANT, AND COPYABLE, GIVING YOU THE MOST FOR YOUR SOFTWARE DOLLARS. DON'T SETTLE FOR LESS.

Beagle Bros™ MICRO SOFTWARE

4315 Sierra Vista, San Diego, Ca 92103
714-296-6400

"APPLE" is a registered trade mark of You-Know-Who.

Apple Mechanic

Shape Writer/Byte-Zap Utility
by Bert Kersey

Another best-selling multiple-utility disk—Nine useful, listable, copyable and customizable programs—

SHAPE EDITOR: Put professional hi-res animation in your programs. Keyboard-draw any shape and let your Apple write a shape table and store it on disk. Design large and small **custom typefaces** too, with special characters. 6 fonts on the disk. LISTable demos show how to use shape tables to animate games, graphic displays, and attractive Charts & Graphs. A valuable time-saving utility/learning tool.

BYTE ZAP: A MUST utility. Rewrite any byte on a disk by loading a sector onto the screen for inspection. **Hex/Dec/Ascii** display optional. Examine bytes via cursor control; enter hex, dec or ascii to change. Create illegal filenames, restore deleted files, change greeting program names, repair/protect disks, change DOS, examine program files. Clear illustrated instructions show how disk data is stored and how to access it. Very educational.

MORE: A disk PACKED with useful music, text and hi-res tricks **for use in your programs.** A great demo-writer program, useful hi-res utilities and educational, entertaining documentation.

$29.50
- ☐ Unprotected disk (48K min.)
- ☐ Beagle Bros Tip Book #5
- ☐ Peeks & Pokes Chart

NEW!

Typefaces

for Apple Mechanic

Here are more hi-res fonts for Apple Mechanic's Xtyper and Hi-Writer programs—26 of them at last count, both large and small, all **proportionally-spaced** and positionable anywhere on either hi-res screen. Most are **full 96-character fonts** many with special graphic characters. Each character (from "!" to "□") of every font (from "Ace" to "Zooloo") is, of course, editable with Apple Mechanic's Font Editor.

BONUS: Here's BEAGLE-MENU! A unique greeting program that displays **only the catalog file names you want** on the screen (for example, only locked-Applesoft files, or only Binary files) for one-key cursor selection. Just hit Return to Run, Brun or Exec the program at the cursor. Many other features— Space-on-Disk, Load/Bload option, forward and backward catalog "scrolling" for easy file location, and optional sector-number elimination. PLUS the ability to **swap file names** in your catalog!

$20.00 Unprotected. Beagle Bros' Apple Mechanic disk is required to utilize the type fonts. Beagle-Menu works with all normal DOS 3.3 disks.

If you don't find our products at your Apple Dealer, tell him to phone Beagle Bros, 714-296-6400, **OR** his favorite software distributor.

NEW!

Flex Text

70-Column Text Utility
by Mark Simonsen

Flex Text is a unique utility that lets you print variable-width text on Apple's hi-res screens in normal 40-column format, 20-column expanded or 56- and **70-column condensed** characters. Character widths may be mixed as you like for emphasis. Flex Text understands **normal Applesoft** Basic commands, including Home, Inverse, Normal, Vtab 1-24 and **Htab 1 through 70**! It also supports text window pokes and scrolling, so you can program normally, but with the **ability to add text to graphics**, or graphics to text! You can even run your existing programs using these features!

FLEX TEXT IS COMPATIBLE WITH DOS TOOL KIT© FONTS

Enter up to NINE font names in Flex Text's boot-up program for easy ctrl-command access. **Upper & lower case** in any character-width without hardware. All characters redefinable with a **text character editor**. Toggle between "normal" text screen and both hi-res pages. Compatible with Neil Konzen's Program Line Editor© and GPLE.©

$29.50 Machine language. Unprotected. 48K min. Peek/Poke chart included. Condensed character display requires a monitor (instead of a tv) for best results.


```
10 REM HI-RES NUMBER GENERATOR
20 SIZE=5: SCALE=SIZE: REM NUMBER-HEIGHT
30 HGR: HOME: POKE 232, 0: POKE 233, 3: ROT=0
40 FOR A=768 TO 830: READ B: POKE A, B: NEXT A
50 N=N+1: N$=STR$(N): X=99: Y=0
60 FOR A=1 TO LEN(N$): HCOLOR=0: DRAW 8 AT
   X, Y: HCOLOR=3: DRAW VAL(MID$(N$, A, 1))
   AT X, Y: X=X+SIZE+SIZE: NEXT A: GOTO 50
70 DATA 20, 0, 24, 0, 27, 0, 31, 0, 35, 0, 39, 0, 44, 0,
   49, 0, 52, 0, 57, 0, 53, 62, 36, 0, 49, 38, 0, 53,
   55, 61, 0, 53, 23, 37, 0
80 DATA 46, 38, 52, 0, 61, 46, 62, 5, 0, 61, 54, 37, 7,
   0, 53, 38, 0, 54, 37, 60, 46, 0, 53, 39, 53, 62, 5, 0
```

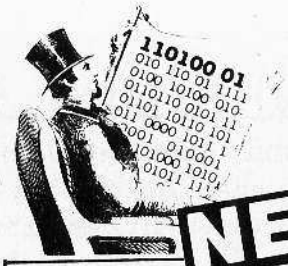
DOS Boss

Disk Command Editor
by Bert Kersey & Jack Cassidy

A classic Apple utility you will ENJOY! Rename DOS commands ("Catalog" can be "Cat", etc.). PROTECT PROGRAMS; any unauthorized save- attempt produces a "Not Copyable" message. Also **List-prevention** and 1-key program-run from catalog. Custom catalogs: Change Disk Volume message to your title; Omit or alter file codes. Rewrite error messages: "Syntax Error" can be renamed "Oops!!" or anything you want! **Two books included**— Fascinating documentation and hours of good Apple reading!

Dos Boss's change features may be appended to your programs so that anyone using your disks (booted or not) formats DOS as YOU designed it.

$24.00
- ☐ Unprotected disk (32K/48K)
- ☐ The Dos Boss Book
- ☐ Beagle Bros Apple Tip Book #2
- ☐ Peeks & Pokes Chart



NEW!

ProntoDOS

High-Speed DOS Utility
by Tom Weishaar

ProntoDos is FAST, saving you time where it counts the most. This comparison with normal Apple-DOS speaks for itself—

FUNCTION	PRONTO	NORMAL
BLOAD HI-RES IMAGE	3-sec.	10-sec.
BSAVE HI-RES IMAGE	6-sec.	12-sec.
LOAD 60-SECTOR PROGRAM	4-sec.	16-sec.
SAVE 60-SECTOR PROGRAM	9-sec.	24-sec.
BLOAD INT/LANGUAGE CARD	4-sec.	13-sec.
TEXT FILES	(no change)	

MORE DISK SPACE: Booting ProntoDos frees up 16 extra sectors of Disk Space, almost a full track. To speed up your Apple, just boot ProntoDos or any disk you have updated with ProntoDos, and you're in business. You can even create new ProntoDos disks with Apple's normal INIT command. ProntoDos is compatible with ALL commands and performs normally (but FAST) with almost ALL programs.

$29.50 Machine Language. Unprotected. Peeks & Pokes chart included. All normal 3.3 disks are updatable

Tip Disk#1

100 Tip Book Tips on Disk
by Bert Kersey

100 programs from Beagle Bros' Tip Books 1, 2, 3 and 4— Fascinating tricks to make your Apple do things it's never done before! All 100 programs are listable, copyable and changeable; and each teaches another fascinating Apple programming technique. **Two different charts are included.**

$20.00
- ☐ Unprotected (32K/48K)
- ☐ Peeks & Pokes Chart
- ☐ Apple II Command Chart

BEAGLE BROS DISKS ARE UNLOCKED AND UNPROTECTED. THIS MEANS EVERY PROGRAM IS INSPECTABLE, CUSTOMIZABLE, IF YOU WANT, AND COPYABLE, GIVING YOU THE MOST FOR YOUR SOFTWARE DOLLARS. DON'T SETTLE FOR LESS.

Beagle Bros™ MICRO SOFTWARE

4315 Sierra Vista, San Diego, Ca 92103
714-296-6400

"APPLE" is a registered trade mark of You-Know-Who.

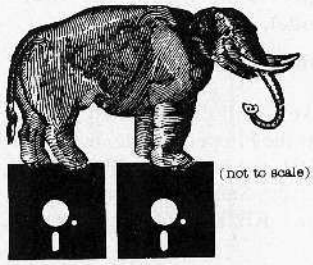
Utility City

21 Utilities on One Disk
by Bert Kersey

A best-seller since it hit the market, and a MUST for your program-development library. Take a look at the features— **List Formatter** makes properly-spaced & indented listings with page breaks; each statement on new line, if-thens and loops called out; a great de-bugger! **Multi-Column Catalog** in any page-width to any printer or CRT. Auto-post Run-Number and last-used Date in programs. Put INVISIBLE working commands in your listings. Access program lines in memory for repair & illegal alteration. Alphabetize & store info on disk. Run any program while another stays intact. Renumber to 65535. Save inverse, trick and **invisible file names**. Convert dec to hex & binary, or Integer to FP. Append programs. Dump text screen to printer...

21 LISTABLE UTILITIES TOTAL!

$29.50
- ☐ Unprotected disk (48K min.)
- ☐ Beagle Bros Apple Tip Book #3
- ☐ Peeks & Pokes Chart


(not to scale)

GOTO Your Apple Dealer.

Most dealers carry our software. If yours doesn't, he can have our disks in his store for you **within 2-3 days** by calling us or his favorite software distributor.

Or Order by Phone:

24-Hour **Toll-Free** Order Desk:
Visa/MasterCard/COD* orders, call
Nationwide: **1-800-854-2003** ext.827
California: **1-800-822-1500** ext.827
Alaska/Hawaii: **1-800-854-2628** ext.827
(ORDERS ONLY, PLEASE) *COD, add $3.00

OR ORDER BY MAIL—

RUSH! The disk packages checked below plus the free 11x17 Peeks & Pokes Chart:

☐ Alpha Plot	$39.50	☐ Frame-Up	$29.50
☐ A.Mechanic	$29.50	☐ ProntoDos	$29.50
☐ Beagle Bag	$29.50	☐ Tip Disk#1	$20.00
☐ DOS Boss	$24.00	☐ Typefaces	$20.00
☐ Flex Text	$29.50	☐ Utility City	$29.50

(Add $1.50 Shipping, any size order. California, add 6% tax.)

NAME$ _____
ADDR$ _____
CITY$ _____
ZIP$ _____

VISA, MASTERCARD Include Card Number & Expiration Date.

Mail U.S check, money order or Visa/MasterCard numbers to:

BEAGLE BROS, DEPT. H
4315 SIERRA VISTA
SAN DIEGO, CA 92103

All Orders Shipped Immediately.

Please add $4.00 for shipping outside North America. COD orders add $3.00. California residents, add 6%.

the user for anything, but is listed in Figure Three for the curious.

Upon running Design Plus, you will first be asked for the increment. You can enter a number or press RETURN for the default value of 2.

After entering the increment, a brief summary of the commands and the prompt 'PRESS RETURN TO CONTINUE' appear. When RETURN is pressed, the program starts drawing.

The commands for Design Plus are easy to understand:

SPACE **Begins the "Step Mode", which allows you to go through the process step-by-step.**

F **Calls the Flipper subroutine (only from within the Step Mode).**

ctrl-C **Exits the program.**

All other keys will exit the Step Mode. Any key exits the Flipper routine, but if the key is a SPACE the program will immediately re-enter the Step Mode (but not the flipper routine). RESET exits the program.

Figure 2

Variable	Value (in hex/dec)	Purpose
DRW	$E6/230	Offset for hi-res pages. The contents of this location determine which page is drawn on by not only DRAW and XDRAW, but HPLOT as well.
D1	$20/32	The value to put in DRW to draw on page 1.
D2	$40/64	The value to place in DRW to draw on page 2.
S1	$CO54/-16300	The location to access to display page 1.
S2	$CO55/-16299	The location to access to display page 2.

Figure 3

800A:AD 54 C0	LDA $C054	Display page 1.
800D:AD 00 C0	LDA $C000	Read the keyboard.
8010:C9 7F	CMP $7F	Has a key been pressed?
8012:10 07	BPL $801B	Yes; then return
1014:EA	NOP	Make the timing right.
8015:AD 55 C0	LDA $C055	Display page 2.
8018:4C 0A 80	JMP $800A	Go thru again.
801B:60	RTS	Return to BASIC.

Design Plus

```
10 B$ =  CHR$ (8): TEXT : GOTO 34
    0
20  REM SET PAGE 1
30  POKE DRW,D1
40  REM DRAW ON 1
50  GOSUB 190
60  REM SHOW PAGE 1
70  POKE S1,0
80  REM SET PAGE 2
90  POKE DRW,D2
100  GOSUB 220
110  REM DRAW ON 2
120  GOSUB 190
130  REM SHOW PAGE 2
140  POKE S2,0
150  REM INCREMENT SCALE
160  GOSUB 220
170  REM READ KEYBOARD
180  GOSUB 250: GOTO 30
190  SCALE= SC
200  XDRAW 1 AT 140,96
210  RETURN
220 SC = SC + INC
230  IF SC > 255 - INC THEN SC =
    1
240  RETURN
250 K% =  PEEK ( - 16384): IF K% =
    160 THEN 270
260  RETURN
270  POKE  - 16368,0
280 K% =  PEEK ( - 16384): IF K% <
    127 THEN 280
290  IF K% <  > 160 THEN  POKE  -
    16368,0
300  IF K% = 198 THEN  POP : GOTO
    320
310  RETURN
320  CALL 32770
330  GOTO 30

340  REM START PROGRAM
350  HOME : PRINT  SPC( 8)"SELECT
     A NUMBER      (0-255)"
360  PRINT : PRINT  SPC( 15)"DEFA
    ULT = 2"
370  VTAB 1: HTAB 27: INPUT A$: PRINT
    A$
380  PRINT : IF A$ = "" THEN A$ =
    "2"
390 INC =  VAL (A$): IF INC = 0 THEN
    340
400  VTAB 10: PRINT
410  PRINT  SPC( 7)"SPACE = ACTIV
    ATE STEP MODE"
420  PRINT
430  PRINT  SPC( 8)" ESC = EXIT S
    TEP MODE"
440  PRINT
450  PRINT  SPC( 9)"  F = FAST FL
    IP!"
460  VTAB 20: HTAB 8
470  PRINT "PRESS "; : INVERSE : PRINT
    "<RETURN>"; : NORMAL : PRINT
    " TO CONTINUE";
480  GET A$
490 DRW = 230:D1 = 32:D2 = 64
500 S2 =  - 16299:S1 =  - 16300
510  REM POKE IN TABLE
520  FOR A = 32768 TO 32795: READ
    B: POKE A,B: NEXT
530  DATA 1,0,4,0,56,54,45,36,7,0
540  DATA 173,84,192,173,80,192,2
    01,127,16,7,234,173,85,192,7
    6,10,128,96
550  REM HI-RES
560  HGR : HGR2
570  REM SET SCALE, ROT
580 SC = 1: ROT= 0
590  REM SHAPE POINTER
600  POKE 232,0: POKE 233,128
610  GOTO 30
620  REM DESIGN PLUS
630  REM BY NEIL TAYLOR
```

A Shape Table Mini-Editor

If you haven't got a shape table editor, and you need to make some vector shapes right now . . . try Faster Shapes, a mini-editor. It's short, fast, and easily typed. It also creates compact shape definitions, a "must" for faster graphics.

VECTOR GRAPHICS

by Enrique A. Gamez

Recently I've become involved with animation—page-flipping, etc.—and found execution time to be a very important factor. I reduced my drawing and movement/logic routines to a minimum, yet that wasn't enough. To make a long story short, I discovered I had been using a hi-res shape-drawing program which wasted much memory space as a tradeoff for convenience of use.

Where speed isn't a consideration, the "block-scan" shape drawing programs like those which use the lo-res screen are very useful. Of course, viewing the lo-res screen, later to be translated to hi-res, gives a distorted image. You may end up with a shape table two or three times longer (thus slower) than necessary in certain circumstances.

The best method, then, is to follow the directions on page 92 of the Programming Reference Manual with graph paper in hand. This is exactly what I've done in the two-part program below. I may not go out much, but I haven't seen any other assembly language program that is quite like this. My entry prompts are patterned after the "Shape Creation Program" on page 216 of the Apple II User's Guide (which has a bug in it).

To enter and save the assembly language program in the usual way:

```
BSAVE A.L.SHAPES, A$3ØØ, L$BB
```

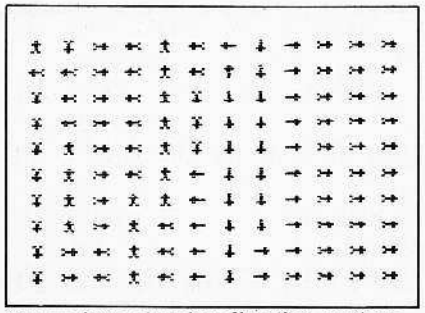
After the Applesoft section of the program BLOADs the machine code, respond to the first question (ENTER DIRECTION VECTOR =) with a U, R, D, or L. The second question requires a Y/N answer. Remember that two "U,N" moves in a row are not allowed. An "F" response to both questions means DONE which sends you into the edit mode. From the graphic display on the upper half of the screen it's easy to find and correct any mistakes which might have been made during entry.

Of course, many possible enhancements could be added, mostly within the Applesoft half. You could provide for a greater number of shapes or for more than the 256 vectors possible. Some nicer editing features could also be added.

I believe the assembly language half has been streamlined to the MAX . . . there's a challenge if ever I heard one!

A.L. Shapes

```
1000 *////////////////////////////////*
1010 * SHAPE TABLE MAKER *************
1020 * ENRIQUE A. GAMEZ  * 7/1982 ****
1030 * BSAVE AS 'A.L. SHAPES' ********
1040 *////////////////////////////////*
1050
1060
1070          .OR $300
1080          .TF A.L. SHAPES
1090
1100
1110 HOLDIT   .EQ $FC
1120 SOURCE   .EQ $FD
1130 OBJECT   .EQ $FE
1140 COUNTER  .EQ $FB
1150 *-------------------------------
1160 * CALCULATE BYTES ******
1170 *-------------------------------
1180 CALCULATE.BYTES
1190          LDA #0
1200          STA SOURCE    OFFSET
1210          STA OBJECT    OFFSET
1220          STA HOLDIT    E.O.R.'ED
1230          LDA #$A
1240          STA COUNTER   ABC'S
1250 *-------------------------------
1260 GO       LDX SOURCE
1270          LDA $6000,X
1280          CMP #$FF
1290          BEQ .99
1300          CMP #0
1310          BEQ .4
1320          CMP #$4
1330          BCS .2
1340          JSR STORE.LOGIC
1350          CPX #$C
1360          BNE .1
1370          LDX OBJECT
1380          STA $7004,X
1390          LDY #0
1400          STY HOLDIT
1410          INC OBJECT
1420 .1       JSR .8
1430          JMP GO
1440 *-------------------------------
1450 .2       LDX COUNTER
1460          CPX #$C
1470          BNE .3
1480          JSR .39
1490          JMP GO
1500 *-------------------------------
1510 .4       LDX COUNTER
1520          CPX #$B
1530          BEQ .5
1540          BCS .39
1550 .3       JSR .7
1560          JMP GO
1570 *-------------------------------
1580 .5       LDX SOURCE
1590          INX
1600          LDA $6000,X
1610          CMP #4
1620          BCS .39
1630          JMP .3
1640 *-------------------------------
1650 .6       LDA HOLDIT
1660          LDX OBJECT
1670          STA $7004,X
1680          RTS
1690 *-------------------------------
1700 .7       JSR STORE.LOGIC
1710 .8       INC SOURCE
1720 .9       INC COUNTER
1730          LDX COUNTER
1740          CPX #$D
1750          BNE .17
1760 .18      LDX #$A
1770          STX COUNTER
1780 .17      RTS
1790 *-------------------------------
1800 .39      JSR .6
1810          LDA #0
1820          STA HOLDIT
1830          INC OBJECT
1840          JSR .18
1850          JMP GO
1860 *-------------------------------
1870 .99      JSR .6
1880          BEQ TOTALLY.DONE
1890          INX
1900          LDA #0
1910          STA $7004,X
1920 TOTALLY.DONE
1930          RTS
1940 *-------------------------------
1950 *:::::::::::::::::::::::::::::::
1960 *-------------------------------
1970 STORE.LOGIC
1980          LDX COUNTER
1990          CPX #$8
2000          BEQ .2        IT'S B
2010          BCS .1        IT'S C
2020          JMP .3        IT'S A
2030 .1       ASL
2040          ASL
2050          ASL
2060 .2       ASL
2070          ASL
2080          ASL
2090 .3       EOR HOLDIT
2100          STA HOLDIT
2110          RTS
2120 *-------------------------------
2130 * ARROW SHAPE TABLE *************
2140 *-------------------------------
2150          .HS 02000600C003C0C
2160          .HS 15D636003C0C15D6
2170          .HS 17005000
2180 *-------------------------------
```



Screen dump showing direction vectors for a miscellaneous shape.

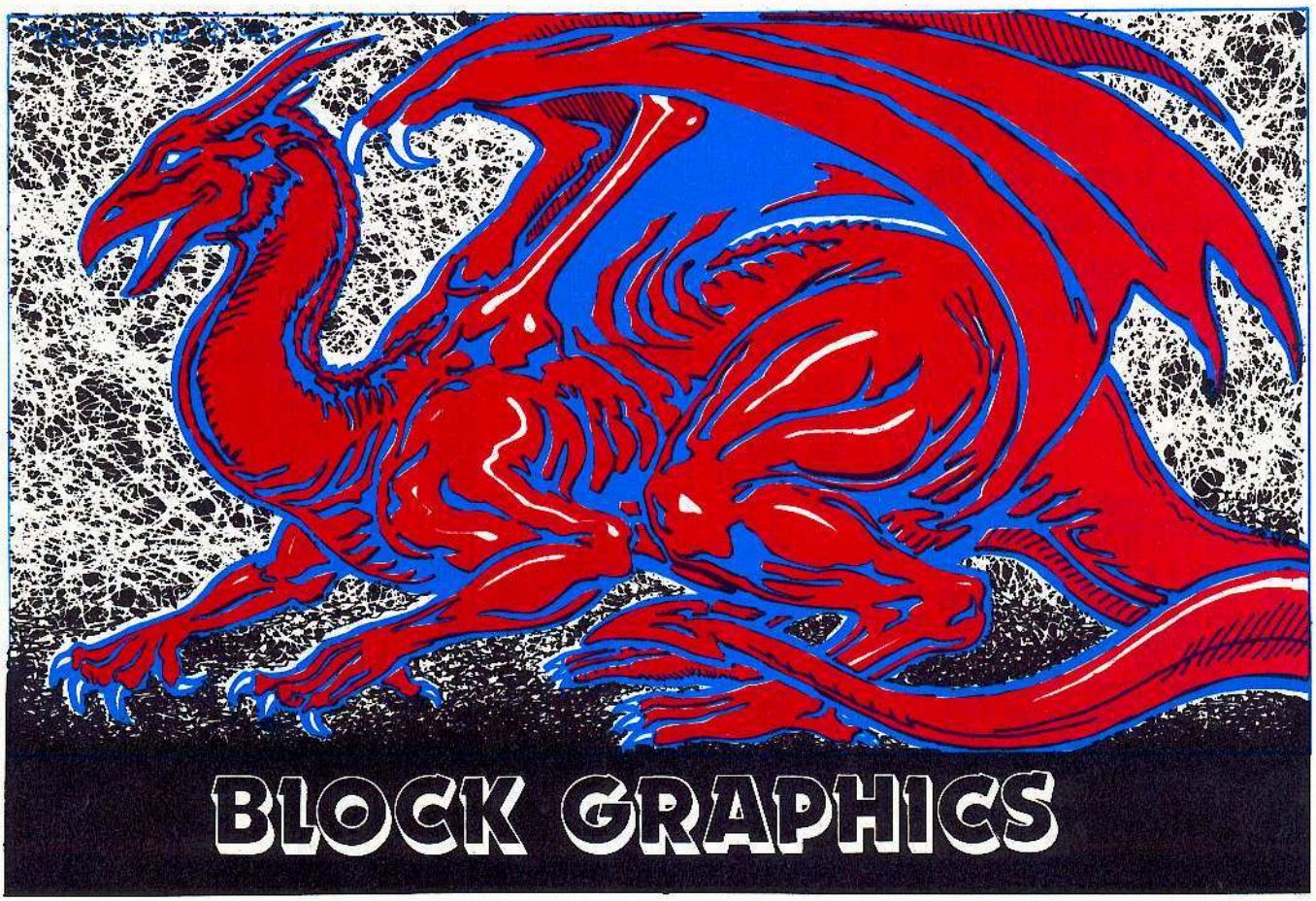
Faster Shapes

```
10  HOME : VTAB 10
20  PRINT "FASTER SHAPES BY ENRIQ
    UE A. GAMEZ"
30 D$ =  CHR$ (4): REM  CTRL D
40  PRINT D$"BLOAD A.L. SHAPES"
50  DIM C$(255),LX(255),LY(255)
60 X = 2:Y =  - 8
70  POKE 232,167: POKE 233,3
80  POKE 28672,2: POKE 28673,0: POKE
    28674,4: POKE 28675,0
90  REM ZERO FLAG
100 ZF = 0
110  HGR : HCOLOR= 3: SCALE= 1: HOME

120  REM
ENTRY

130  FOR CN = 0 TO 255
140  VTAB 22: CALL  - 958: VTAB 2
    3: INVERSE : PRINT "F,F TO Q
    UIT": NORMAL
150  VTAB 22
160  PRINT "DIRECTION VECTOR #";C
    N + 1" ? ";: GET A$: PRINT A
    $
170  CALL  - 868: PRINT "PLOT OR
    NOT? (Y/N)";: GET B$: PRINT
    B$
180 C$(CN) = A$ + B$
190  GOSUB 730
200  IF (XX > 0 AND XX < 10) THEN
    ZF = 0: GOTO 230
210  IF (XX = 0 AND ZF = 1) THEN
     PRINT  CHR$ (7): HOME : VTAB
    21: INVERSE : PRINT "YOU CAN
     NOT ENTER U,N": NORMAL : GOTO
    150
220  IF XX = 0 THEN ZF = 1: GOTO
    260
230  IF XX = 30 THEN  GOTO 150
```

Hex Dump for A.L. Shapes

```
0300- A9 00 85 FD 85 FE 85 FC
0308- A9 0A 85 FB A6 FD BD 00
0310- 60 C9 FF F0 70 C9 00 F0
0318- 28 C9 04 B0 18 20 91 03
0320- E0 0C D0 0B A6 FE 9D 04
0328- 70 A0 00 B4 FC E6 FE 20
0330- 67 03 4C 0C 03 A6 FB E0
0338- 0C D0 0E 20 76 03 4C 0C
0340- 03 A6 FB E0 0B F0 08 B0
0348- 2D 20 64 03 4C 0C 03 A6
0350- FD E8 BD 00 60 C9 04 B0
0358- 1D 4C 49 03 A5 FC A6 FE
0360- 9D 04 70 60 20 91 03 E6
0368- FD E6 FB A6 FB E0 0D D0
0370- 04 A2 0A 86 FB 60 20 5C
0378- 03 A9 00 85 FC E6 FE 20
0380- 71 03 4C 0C 03 20 5C 03
0388- F0 06 E8 A9 00 9D 04 70
0390- 60 A6 FB E0 0B F0 08 B0
0398- 03 4C A2 03 0A 0A 0A 0A
03A0- 0A 0A 45 FC 85 FC 60 02
03A8- 00 06 00 0C 00 3C 0C 15
03B0- D6 36 00 3C 0C 15 D6 17
03B8- 0D 05 00
```

```
240  IF XX = 20 THEN  PRINT  CHR$
     (7): INVERSE : PRINT  CHR$ (
     7): INVERSE : PRINT "NOT VAL
     ID COMMAND - USE F,F TO QUIT
     ": NORMAL : GOTO 150
250  IF XX = 10 THEN TT = CN: GOTO
     340
260  POKE (24576 + CN),XX
270 Y = Y + 10
280  IF Y > 95 THEN Y = 2:X = X +
     10
290 LX(CN) = X:LY(CN) = Y
300  DRAW SH AT X,Y
310  HOME
320  NEXT CN
330  REM
EDIT
340  HOME
350  VTAB 22
360  PRINT " CHANGE WHICH 1-";TT;
     " VECTOR (0=END) ";: INPUT C
     N
370  IF CN = 0 THEN 650
380  IF CH > TT THEN  HOME : VTAB
     21: INVERSE : PRINT "VECTOR"
     ;CN;" DOES NOT EXIST": NORMAL
     : GOTO 350
390 CN = CN - 1
400 PK =  PEEK (24576 + CN)
410  IF PK < 4 THEN SH = 1: ROT=
     16 * PK: GOTO 430
420 SH = 2: ROT= 16 * (PK - 4)
430  FOR TC = 0 TO 3
440  FOR TM = 0 TO 150: NEXT TM
450  XDRAW SH AT LX(CN),LY(CN)
460  FOR TM = 0 TO 150: NEXT TM
470  DRAW SH AT LX(CN),LY(CN)
480  NEXT TC
490  REM
CHANGE VECTOR

500  HOME
510  VTAB 23: INVERSE : PRINT "X,
     X TO QUIT": NORMAL
520  VTAB 22
530  PRINT "ENTER DIRECTION VECTO
     R #";CN + 1" ? ";: GET A$: PRINT
     A$
540  PRINT "PLOT OR NOT? (Y/N)";:
      GET B$: PRINT B$
550 C$(CN) = A$ + B$
560  XDRAW SH AT LX(CN),LY(CN)
570  GOSUB 730
580  HOME
590  VTAB 21
600  IF (XX = 0 AND ( PEEK (24576
      + (CN - 1)) = 0 OR  PEEK (2
     4576 + (CN + 1)) = 0)) THEN
      FLASH : PRINT "ALERT:";: INVERSE
     : PRINT " YOU HAVE CONSECUTI
     VE ZEROES!": NORMAL
610  POKE (24576 + CN),XX
620  DRAW SH AT LX(CN),LY(CN)
630  GOTO 350
640  REM
DISPLAY SHAPE
650  CALL 768: PRINT "YOUR TABLE
     STARTS AT $7000 ... ENJOY!"
660  PRINT : INPUT "DO YOU WANT T
     O SEE IT (Y/N)? ";A$
670  IF A$ = "N" THEN 710
680  REM $7000=28672
690  HGR : POKE 232,0: POKE 233,1
     12
700  DRAW 1 AT 140,80
710  END
720  REM
INTERPRET VECTOR
730  IF C$(CN) = "UN" THEN XX = 0
     : ROT= 0:SH = 1: RETURN
740  IF C$(CN) = "RN" THEN XX = 1
     : ROT= 16:SH = 1: RETURN
750  IF C$(CN) = "DN" THEN XX = 2
     : ROT= 32:SH = 1: RETURN
760  IF C$(CN) = "LN" THEN XX = 3
     : ROT= 48:SH = 1: RETURN
770  IF C$(CN) = "UY" THEN XX = 4
     : ROT= 0:SH = 2: RETURN
780  IF C$(CN) = "RY" THEN XX = 5
     : ROT= 16:SH = 2: RETURN
790  IF C$(CN) = "DY" THEN XX = 6
     : ROT= 32:SH = 2: RETURN
800  IF C$(CN) = "LY" THEN XX = 7
     : ROT= 48:SH = 2: RETURN
810  IF C$(CN) = "FF" THEN  POKE
     (24576 + CN),255:XX = 10: RETURN
820  IF C$(CN) = "XX" THEN XX = 2
     0: RETURN
830  PRINT  CHR$ (7): HOME : VTAB
     21: INVERSE : PRINT "WHAT DI
     D YOU SAY?": NORMAL :XX = 30
     : RETURN

840  REM
COPYRIGHT 1983

850  REM
SOFTKEY PUBLISHING
```



The two major advantages of text and lo-res graphics are their simple commands and the relatively faster graphic display capability. Entire areas can be colored with simple PRINT or PLOT (VLIN, HLIN) commands. Unfortunately, detail (resolution) is exchanged for speed and ease of use. Hi-res shapes, especially complex ones, are both slow and laborious using HPLOTS or shape tables. Programs that require detailed but complex shapes, yet have to be fast (like arcade games) can only be created with a great deal of programming knowledge beyond BASIC languages. When talking about fast graphics, one usually is talking about "machine speed", and that usually means a program written in machine language.

Yet there is a way to combine the speed of text and lo-res with the detail of hi-res without knowing a bit of machine or assembly language. But you have to use routines that are already written in machine language, usually by an experienced assembly language programmer, preferably one well-versed in the eccentricities of hi-res page formatting.

One of the best methods is to use a program called a "character generator." The name is misleading. It can be used to place characters on the hi-res image area. But it can also be used to place other images there. An example of a hi-res character generator is Apple's ***Tool Kit.*** There are many such programs now available. With it, a BASIC programmer can create arcade quality games and demos on the hi-res page by using simple text PRINT statements.

CORE presents its own character generator: ***Quick Draw,*** by Robb Canfield.

Unlike other generators that use only 64 or 128 characters per set, programmers who use Robb's ***Quick Draw*** can print the entire set of 256 characters by specifying INVERSE, FLASH, or NORMAL. Control and lowercase sections can also be used. With his character editor, all 256 characters can be filled with your own assortment of shapes. And by specifying the size of the shape (how many characters tall and wide it is), it is easy to print a huge shape (let's say one that's six characters wide and five tall) with just a simple:

PRINT "A"

instead of writing:

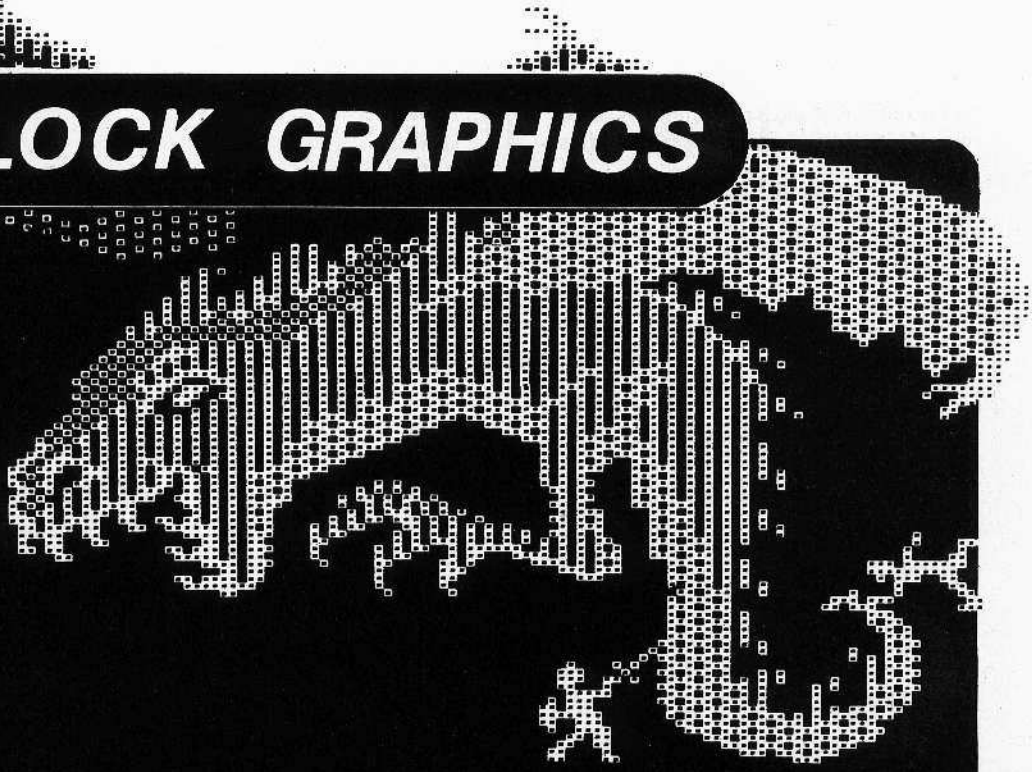
PRINT "ABCDEF"
PRINT "GHIJKL"
PRINT "MNOP"
PRINT "STUVWX"

Character generators make it easy for a BASIC programmer to write fast and detailed graphic presentations because it becomes as easy as a simple PRINT.

(We encourage you to use this program for your own non-commercial software. And if you've written one that you feel is worth marketing commercially, submit it to Softkey for publication in CORE and on disk.)

BLOCK GRAPHICS

A special feature for BASIC programmers



Arcade Quality Graphics

by Robb Canfield

REQUIREMENTS:
48K Apple II (or Franklin Ace 1000)
One disk drive.

Games seem to have universal appeal. From the toddlers to the elderly, all seem to enjoy becoming a hero in a dungeon or a space pilot in the remoteness of the universe. Arcade games are the most popular with their high-speed animation and colorful creatures. Unfortunately, the techniques of displaying shapes quickly and easily have not been revealed to most programmers.

Finding a way to quickly draw and move shapes is the biggest problem you will encounter when designing a game. In this article, I will concentrate on drawing shapes easily and quickly. So I introduce my Quick Draw 1.0, a Hi-Res Character Generator (HCG).

One method of displaying shapes on the hi-res screen is to use shape tables or an equivalent. Shape tables use a vector approach to draw a shape on the screen. Vectors show which direction to move and whether to plot a point or not. This method has many benefits. Enlarging a shape (SCALE) and rotating a shape (ROT) are easily done. Even changing the color becomes a simple exercise.

Another advantage of shape tables is their ability to be placed anywhere on the hi-res screen. Since the shape is stored as vectors, it really makes no difference where the shape is drawn. The disadvantage of shape tables is that they are very, very slow.

An HCG usually does not handle such exotic functions as ROTate and SCALE, but shapes are drawn very quickly. Shapes for HCG are not stored as vectors. Instead, they are stored as bytes of pure data. A fair analogy would be to compare shape tables to walking from one place to another. There are many routes one could choose to get there, but it is time-consuming. A HCG, in comparison, would be our futuristic teleporter that instantly transports us from point A to point B. We see less, but get there quicker.

Another limiting feature about a HCG is its inability to draw an object anywhere on the hi-res screen. As the HCG stands now, it can only display shapes on 24 rows and 40 columns, much like the text screen. And that is why it is so easy to use.

I have developed Quick Draw 1.0 so that Applesoft, Integer, and machine language can call it simply by PRINTing a character. However, I have not provided any scroll routines for the hi-res screen. This means that once you are at the bottom of the screen, your shapes will overwrite each other (so be very careful).

Putting the HCG into Memory

1) Enter the monitor.
CALL -151
2) Enter the program for Quick Draw 1.0. Remember to press the RETURN key at the end of every line.
3) Change the Applesoft program pointer to $6001.
67:01 60
6000:00 00 00
4) Return to BASIC.
3D0G
5) Save the HCG.
BSAVE QUICK DRAW,A$800,L$89
6) Make sure everything is okay.
NEW
7) Type the BASIC program, MAKE TABLES. Then run it. This program will automatically create and save all the tables for Quick Draw 1.0.

The Editor

As I mentioned earlier, shapes for a HCG are stored as bytes in memory. These bytes are moved from the "Character Set" to the hi-res screen when needed. A "character"

is 7 pixels across and 8 pixels high. A shape can be any number of these characters placed together.

I have provided an editor for drawing and manipulating shapes easily.

Commands featured are:

- Inverse
- Change high bit
- Continuous draw/undraw
- Some other ordinary commands.

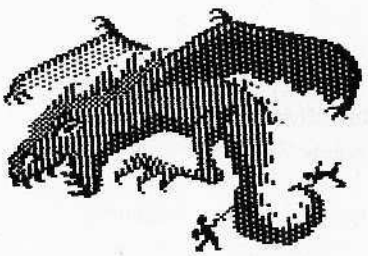
To use the editor, two machine language programs must be entered, along with the BASIC program and the Quick Draw editor's "Character Set".

WARNING: There is no DOS error-handling, so if a DOS error occurs you will exit the program. If this happens press RESET and type RUN. The "Character Set" in memory will be unharmed, but any shape you were editing will have been lost.

Quick Draw

```
1000 ***************************
1010 *                         *
1020 *    HI-RES CHARACTER     *
1030 *       GENERATOR         *
1040 *         VER XII         *
1050 *     QUICK DRAW 1.0      *
1060 *                         *
1070 *           BY            *
1080 *                         *
1090 *    ROBB S. CANFIELD     *
1100 *                         *
1110 *     COPYRIGHT 1982      *
1120 *                         *
1130 *   SOFTKEY PUBLISHING    *
1140 *                         *
1150 ***************************
1160
1170 ********************************
1180 *                              *
1190 * THIS VERSION DOES NOT WRITE  *
1200 * TO THE TEXT SCREEN. HAS TEXT *
1210 * WINDOW AND ADVANCES CH AND   *
1220 * CV WHEN NECESSARY.           *
1230 *                              *
1240 * $06 - HORIZONTAL BLOCKS (1)  *
1250 * $07 - VERTICAL BLOCKS (1)    *
1260 * $04,$05 - LOCATION OF        *
1270 *           CHARACTER SET      *
1280 *                              *
1290 * THIS VERSION IS LOCATED AT   *
1300 * $803 FOR THE EDITOR.         *
1310 *                              *
1320 ********************************
1330
1340
1350 *
1360 * LOCATIONS USED IN THE PROGRAM
1370 *
1380
1390 WNDWDTH  .EQ $21      WIDTH OF TEXT WINDOW
1400 WNDBTM   .EQ $23      BOTTOM OF TEXT WINDOW
1410
1420 INV.FLAG .EQ $32      THE INVERSE FLAG. USED FOR FLASH AND INVERSE
1430
1440
1450 YSAVE    .EQ $D0      YREG SAVE AREA
1460 XSAVE    .EQ $D1      XREG SAVE AREA
1470
1480 ACCUM    .EQ $D2      ACCUMULATOR SAVE
1490 VERTSAVE .EQ $D3      VERTICAL POSITION SAVE AREA
1500 CHARLOC  .EQ $4       LOCATION OF CHARACTER SET ($4 & $5)
1510 SPOT     .EQ $D4,$D5  LOCATION TO PLACE CHARACTER ON HI-RES SCREEN
1520 CV       .EQ $25      CURSOR VERTICAL POSITION
1530 HIRES.PAGE .EQ $E6    PAGE WE ARE ON ($20=PAGE 1, $40=PAGE 2)
1540 CH       .EQ $24      CURSOR HORIZONTAL POSITION
1550 HORZ2    .EQ $8       BLOCKS HORIZONTALLY
1560 TEMP     .EQ $9       TEMPORARY STORAGE FOR ANYTHING
1570 HORZ     .EQ $6       NUMBER OF VERTICAL LINES TO PRINT
1580 VERT     .EQ $7       NUMBER OF LINES IN BLOCK
1590 VERT2    .EQ $0       LINES TO PRINT (8 * VERT)
1600 CHAR     .EQ $01      CHARACTER WE ARE ON (USED FOR BLOCK SIZES)
1610          .OR $803
1620          .TF QUICK DRAW.OBJ
1630
1640 *
1650 * SET FOR HOOK UP WITH PRINT
1660 *
1670
1680          STA ACCUM    SAVE ALL THE REGISTERS
1690          STX XSAVE
1700          STY YSAVE
1710          LDX HORZ     SAVE BLOCK SIZE
1720          STX HORZ2
1730          LDX VERT
1740          LDY VERTICAL,X
1750          STY VERT2
1760 CHARTBL1 CMP #$8D
1770          BEQ CR
1780          AND INV.FLAG USE FLASH/INVERSE MODES
1790 DRAW     TAY
1800          STY CHAR
1810          CLC
1820          LDA CHRLOW,Y GET LOW BYTE OFFSET
1830          ADC CHARLOC  AND ADD TO CHARLOC
1840          STA GET+1
1850          LDA CHRHIG,Y ADD ON HIGH BYTE
```

```
1860          ADC CHARLOC+1
1870          STA GET+2
1880          LDX CV        FIND VERTICAL POSITION ON HI-RES SCREEN
1890          LDY VERTICAL,X
1900          STY VERTSAVE
1910          LDX #$00      RESET LOOP FOR CHARACTER
1920 PUTCHAR  CPY #192      OFF SCREEN BOTTOM/TOP
1930          BGE SKIP      YES SO DON'T DRAW
1940          LDA TABLELOW,Y GET ACTUAL LOCATION TO PUT CHARACTER
1950          STA SPOT
1960          LDA TABLEHIG,Y
1970          CLC
1980          ADC HIRES.PAGE
1990          STA SPOT+1
2000          LDY CH        GET HORIZONTAL POSITION
2010 GET      LDA $FFFF,X   GET CHARACTER
2020          STA (SPOT),Y
2030          STA (SPOT),Y
2040 SKIP     INC VERTSAVE GET NEXT LINE ON SCREEN
2050          LDY VERTSAVE
2060          INX           DONE?
2070          CPX VERT2
2080          BNE PUTCHAR   DONE? NO SO CONTINUE
2090          DEC HORZ2
2100          BEQ GOODBYE
2110          LDY CH
2120          INY
2130          CPY WNDWDTH
2140          BEQ CR
2150          STY CH
2160          CLC
2170          LDA VERT      INCREMENT CHARACTERS
2180          ADC CHAR
2190          JMP DRAW
2200 GOODBYE  INC CH        ADVANCE ROUTINE
2210          LDY CH
2220          CPY WNDWDTH
2230          BCC GOODBYE2
2240 CR       LDY #$00      CARRIAGE RETURN CONTROLLER
2250          STY CH
2260          INC CV
2270          LDY CV
2280          CPY WNDBTM
2290          BCC GOODBYE2
2300          DEC CV
2310 GOODBYE2 LDA CHAR
2320          LDX XSAVE
2330          LDY YSAVE
2340          RTS
```



Steps for Entering the Editor

1) Reset Applesoft to its original location.

FP

2) Type the BASIC program, QD.EDITOR, and SAVE it.

SAVE QD.EDITOR

3) Enter the monitor and type the first machine language program.

QD.EDITOR.UTIL.OBJ, and save it.

BSAVE QD.EDITOR.UTIL.OBJ, A$300,L$8A

4) Type the second program,GET.OBJ, and save it also.

BSAVE GET.OBJ,A$1700, L$A3

5) Type the shapes for the cursor into memory and save it.

BSAVE SHAPES,A$300,L$11

6) Enter the editor "Character Set" and save it.

BSAVE EDITOR.SET,A$D00,L $FF

7) Return to BASIC.

8) Type in the BASIC program, MAKE START, and RUN it. This program creates a text file that will load all the files necessary for the Quick Draw editor to work.

9) To run the editor, EXEC the file START.

EXEC START

When the editor is up and running, you will be asked to select the "Horizontal Block Size" and the "Vertical Block Size". Block sizes are measured by how many "Characters" wide or tall the shape will be. A shape is normally the size of one character, but if you set the block size to 3 characters by 2 characters then the shape consists of 6 characters. That's 21 pixels horizontally and 16 pixels vertically. The default answer (if you press RETURN instead of a number) to both the "Horizontal Block Size" and "Vertical Block Size" is 1. The maximum size of any block is 7 characters across and 4 high.

After that you will be asked whether you want to create a shape on a white background instead of a black one. The default answer is "NO" (selecting a black background).

The final question asked is whether sound should be used. I use the Apple's bell occasionally. If this sound is annoying, it can be disabled by typing "N". The default answer is "YES".

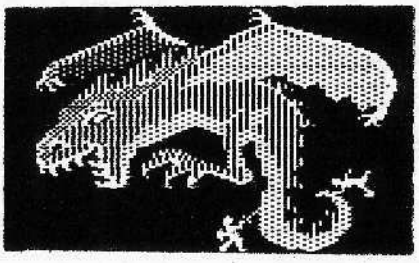
To Create Shapes

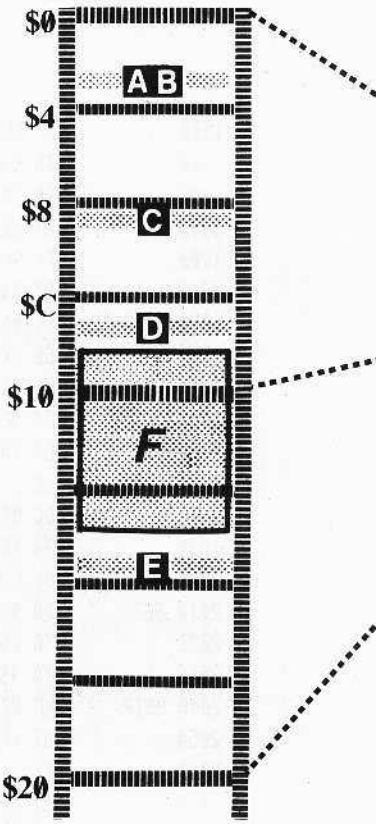
The first thing you should notice is a large box in the center of the screen. This is your Editing Box. Above and to the right of it is the Block Map, consisting of one or

continued on page 44

Make Tables

```
10  REM
MAKE TABLES
20  IF  PEEK (103) = 1 AND  PEEK
    (104) = 64 THEN 40
30  POKE 103,1: POKE 104,64: POKE
    16384,0: PRINT  CHR$ (4)"RUN
    MAKE TABLES"
40  HOME
50  VTAB 2: HTAB 6
60  PRINT "MAKING VERTICAL OFFSET
    TABLE"
70  FOR Y = 0 TO 23
80  POKE 2188 + Y,Y * 8
90  VTAB 12: HTAB 10
100  PRINT Y" "
110  NEXT
120  HOME
130  VTAB 2: HTAB 5
140  PRINT "MAKING CHARACTER OFFS
    ET TABLE"
150  FOR Y = 0 TO 255
160  POKE 2212 + Y,Y * 8 -  INT (
    Y * 8 / 256) * 256
170  POKE 2468 + Y,Y * 8 / 256
180  VTAB 12: HTAB 10
190  PRINT Y"   "
200  NEXT
210  HOME
220  VTAB 2: HTAB 8: PRINT "MAKIN
    G HI-RES LINE TABLE"
230  DEF  FN MOD(A) =  INT ((A /
    8 -  INT (A / 8)) * 8 + .05)
     *  SGN (A / 8)
240 YL = 2724:YH = YL + 192
250  FOR Y = 0 TO 191
260 A =  FN MOD(Y)
270 B =  FN MOD(Y / 8)
280 C =  INT (Y / 64)
290 YA = A * 1024 + B * 128 + C *
    40
300  POKE YH + Y,YA / 256
310  POKE YL + Y,YA -  INT (YA /
    256) * 256
320  VTAB 12: HTAB 10: PRINT Y"
    "
330  NEXT
340  PRINT  CHR$ (4)"BSAVE TABLES
    ,A$88C,L$398"
350  TEXT
```





Make Start *(Step Eight)*

```
10  HOME
20 D$ =  CHR$ (4)
30  PRINT D$"NOMONCIO"
40  PRINT D$"OPEN START"
50  PRINT D$"DELETE START"
60  PRINT D$"OPEN START"
70  PRINT D$"WRITE START"
80  PRINT "POKE 103,1:POKE 104,96
    :POKE 24576,0"
90  PRINT "NEW"
100  PRINT "BLOAD QUICK DRAW.OBJ"
110  PRINT "BLOAD TABLES"
120  PRINT "BLOAD QD.EDITOR.UTIL.
    OBJ"
130  PRINT "BLOAD GET.OBJ"
140  PRINT "BLOAD EDITOR.SET,A$D0
    0"
150  PRINT "BLOAD SHAPES"
160  PRINT "RUN QD.EDITOR"
170  PRINT D$"CLOSE"
180  PRINT D$"NOMONCIO"
```

Commands for Quick Draw

```
  W          UP
A   S    LEFT   RIGHT        ---->   Set the high bit.
  Z          DOWN             <----   Clear the high bit.
```

ctrl I	Inverse shape.
ctrl F	Flip high bit for each segment.
ctrl D	Save or load a character set.
ctrl @	Clear entire character set.
ctrl X	Exit the editor.
G	Get character from Character Set.
P	Put character into Character Set.

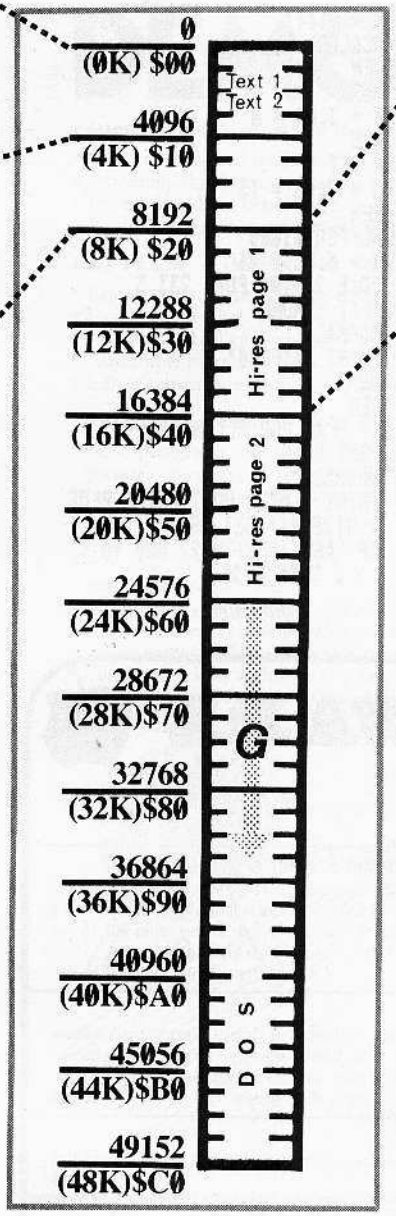
"	Select a new block size.	1	Continuous Plot mode.
!	Erase character.	2	Continuous Erase mode.
ESC	Exit the above options.	3	Special Reverse mode.

Get.Obj *(Step Four)*

```
1700- A4 0B A6 00 86 FE B9 00
1708- 18 08 29 7F A6 FD E0 08
1710- B0 0D CA 4A CA 10 FC A6
1718- FE 86 00 A2 01 D0 02 A2
1720- 07 20 99 17 4A 85 08 B0
1728- 11 A2 00 20 8D 17 A0 03
1730- A2 06 A9 00 20 01 F6 4C
1738- 4F 17 A2 03 20 8D 17 A0
1740- 03 A2 06 A9 00 20 01 F6
1748- 28 08 30 03 A2 00 2C A2
1750- 03 20 8D 17 A0 03 A2 0E
```

Shapes *(Step Five)* — A

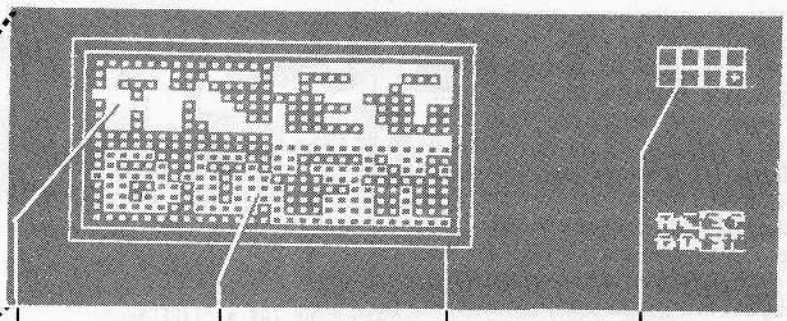
```
0300- 02 00 06 00 0E 00 23 2D
0308- 35 36 3F 27 04 00 35 27
0310- 00
```



E

```
1758- A9 00 20 01 F6 20 9E 17
1760- 18 A5 00 69 04 85 00 A5
1768- 08 CA D0 B5 28 18 A5 01
1770- 69 04 C5 02 90 0C F0 0A
1778- A6 03 86 01 C8 C4 0C D0
1780- 85 60 85 01 38 A5 00 E9
1788- 1C 85 00 D0 EF 20 EC F6
1790- A5 01 A0 00 A6 00 4C 11
1798- F4 86 09 84 0A 60 A6 09
17A0- A4 0A 60
```

F—Character Set Being Edited
G—QD.Editor (Applesoft)



B — *QD.Editor.Util.Obj* *(Step Three)*

```
0311- A4 0B A9 00 99 FF 17      0350- 18 A6 19 F0 04 09 80 30
0318- 88 D0 FA 60 A4 0B B9 FF   0358- 02 29 7F 99 00 18 C8 C4
0320- 17 91 1A 88 D0 F8 60 A4   0360- 0C D0 EB 60 A4 0B B9 00
0328- 0B B1 1A 99 FF 17 88 D0   0368- 18 49 7F 99 00 18 C8 C4
0330- F8 60 A9 0E 8D 3F 03 A9   0370- 0C D0 F3 60 A4 0B B9 00
0338- 00 A2 08 A0 00 8D 00 0E   0378- 18 30 04 09 80 30 02 29
0340- 88 D0 FA CA F0 05 EE 3F   0380- 7F 99 00 18 C8 C4 0C D0
0348- 03 10 F0 60 A4 0B B9 00   0388- ED 60
```

C — *Quick Draw.Obj*

```
0803- 85 D2 86 D1 84            0848- A4 24 BD FF FF 91 D4 91
0808- D0 A6 06 86 08 A6 07 BC   0850- D4 E6 D3 A4 D3 E8 E4 00
0810- 8C 08 84 00 C9 8D F0 5F   0858- D0 DD C6 08 F0 11 A4 24
0818- 25 32 A8 84 01 18 B9 A4   0860- C8 C4 21 F0 12 84 24 18
0820- 08 65 04 8D 4B 08 B9 A4   0868- A5 07 65 01 4C 1A 08 E6
0828- 09 65 05 8D 4C 08 A6 25   0870- 24 A4 24 C4 21 90 0E A0
0830- BC 8C 08 84 D3 A2 00 C0   0878- 00 84 24 E6 25 A4 25 C4
0838- C0 B0 16 B9 A4 0A 85 D4   0880- 23 90 02 C6 25 A5 01 A6
0840- B9 64 08 18 65 E6 85 D5   0888- D1 A4 D0 60
```

D — *Editor.Set* *(Step Six)*

```
0D00- 7F 7F 7F 7F 7F 7F 7F 7F   0D80- 1E 52 52 1E 1E 52 52 1E
0D08- 7F 01 01 01 01 71 11 11   0D88- 3C 25 25 3C 3C 25 25 3C
0D10- 7F 00 00 00 00 7F 00 00   0D90- 78 4B 4B 78 78 4B 4B 78
0D18- 7F 40 40 40 40 47 44 44   0D98- 70 16 16 70 70 16 16 70
0D20- 44 44 44 44 44 44 44 44   0DA0- 61 2D 2D 61 61 2D 2D 61
0D28- 44 44 47 40 40 40 40 7F   0DA8- 43 5A 5A 43 43 5A 5A 43
0D30- 00 00 7F 00 00 00 00 7F   0DB0- 07 34 34 07 07 34 34 07
0D38- 11 11 71 01 01 01 01 7F   0DB8- 7F 7F 00 00 7F 7F 00 00
0D40- 11 11 11 11 11 11 11 11   0DC0- 7F 7F 00 00 7F 7F 00 00
0D48- 00 66 66 00 00 66 66 00   0DC8- 7F 7F 00 00 7F 7F 00 00
0D50- 00 4C 4C 00 00 4C 4C 00   0DD0- 7F 7F 00 00 7F 7F 00 00
0D58- 00 19 19 00 00 19 19 00   0DD8- 7F 7F 00 00 7F 7F 00 00
0D60- 00 33 33 00 00 33 33 00   0DE0- FF FF 00 00 FF FF 00 00
0D68- 7F 41 41 41 41 41 41 7F   0DE8- FF FF 00 00 FF FF 00 00
0D70- 00 00 08 1C 08 00 00 00   0DF0- FF FF 00 00 FF FF 00 00
0D78- 0F 69 69 0F 0F 69 69 0F   0DF8- FF FF 00 00 FF FF 00 00
```

more small square boxes linked together. Each of these smaller boxes represents one character (block) within your current shape. The "+" sign marks which character you are currently in. Below the Block Map is the actual image you are editing. This is an exact representation of your shape, and it will change as you edit its larger look-alike in the Edit Box.

Within the Edit Box are many small squares. These represent one pixel (on the hi-res screen) of the actual image. The blinking box in the upper left-hand corner is your Editing Cursor. It can be moved by pressing W,A,S or X (see "Commands" chart). When a point is plotted (with the SPACE bar), the small square will become noticeably larger. If the center is black, then the high bit (color bit) of that segment is off. If white, then the color bit is on.

There is room for up to 256 characters in the table. The actual number of shapes will vary depending on the number of characters each one takes up. For example, if a shape has a block size of 2 by 4, 8 consecutive characters will be used to store the shape.

continued on page 46

QD.Editor *(Step Two)*

```
10  REM
MO=MODE, EN=ENDING BYTE,
    ST=STARTING BYTE
20 MO = 25:ST = 11:EN = 12:BI = 2
    53
30 H = 1:V = 1
40 I$ = "N":S$ = "Y":I = 128:T$ =
    "IJKL"
50 TA$ = "IIFFCNNL"
60  ONERR  GOTO 3490
70  REM
TABLE FOR C$
80 D$ =  CHR$ (4)
90  REM
CURRENT CHARACTER SET AT
    $300
100 LO = 3584
110  ROT= 0: SCALE= 1
120  REM
INCREMENT FOR VERTICAL C
    HARACTER COUNT
130 IV = .125
140  REM
BUFFER FOR CHARACTER ($1
    800)
150 H6 = 6144
160  HCOLOR= 3
170  REM
HCB
180 N3 = 3:N4 = 8
190  REM
EDITOR SET
200 N5 = 0:N6 = 13
210  REM
WORK BUFFER $1800
220 N7 = 0:N8 = 24
230  POKE 232,0: POKE 233,3
240  TEXT : HOME
250  NORMAL
260  PRINT D$"PR#0"
270  HOME
280  REM
SET BIT FLAG FOR ALL BITS
290  POKE BI,8
300  NORMAL
310  PRINT "ENTER HORIZONTAL BLOC
    K SIZE ";: GET NS$
320  IF  ASC (NS$) = 27 AND TA <
     > 0 THEN 2330
```

continued on page 46

TRY BEFORE YOU BUY.....
DEMO DISK FOR $15.00

Fastest by Far – *Compare speed to any other Apple II word processor . . .*

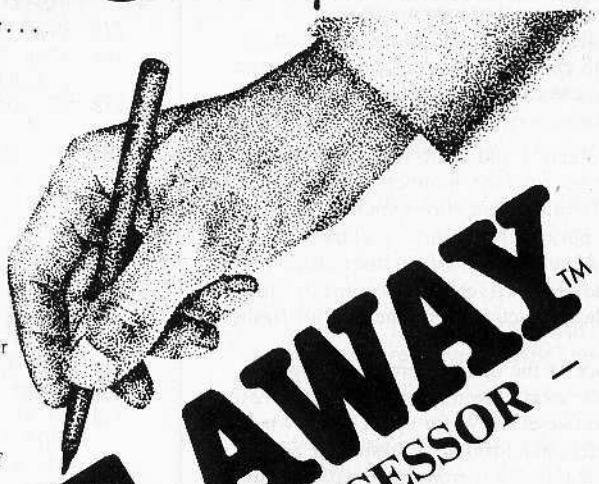
- Text editing speed rivals that of larger computers
- Extremely rapid text formatting
- Instant switching between editor and formatter with language card
- Rapid loading of standard text files

Easy to Use – *designed with YOU in mind . . .*

- Easy to learn – includes interactive Tutorial
- Easy to remember, abbreviated commands
- Cut/Paste/Replicate with ease
- Adjustable automatic word wrap
- Editor remembers document name for easy disk updates

Powerful, Professional Features – *won't soon be outgrown . . .*

- Auto-paragraph, headers, footers, test-page, user-definable page number format, centering, full margin control at all times, incremental spacing, left and/or right justify, tabs and tab stop setting, many more formatting features!
- Edit Macro – much more powerful than a mere "search and replace"!
- File chaining via unique "CALL" feature allows unattended printing of very large documents.
- Form Letter/Mail Merge – ideal for generating form letters, standard legal documents, bulk printing of letters or customized advertisements, begin/repeat blocks, and much more!
- Conditional Text with logical operators adds even more flexibility to Form Letter/Mail Merge feature.



APPLE II WRITE AWAY™
– AN ADVANCED WORD PROCESSOR –

by

Doug Stinson

Complete System $175.00
Includes Full System Plus Extensive Users Manual.

Here's what some of our customers have to say about our word processor:

– *"This is my third (and by far the best) word processing system. This will replace over $300 in other software packages." – J. B., Fenton, Missouri*

– *"The best word processor package for the Apple II that we have evaluated. . . Congratulations on a fine job of software development." – A. C., Charlottesville, Virginia*

– *"Congratulations! I've thoroughly explored it and it's a very fine piece of work - extremely powerful, and, despite my best efforts, virtually crashproof." – J. B., St. Louis, Missouri*

– *"I love your word processor" – H. S., Denver, Colorado*

RECOMMENDED ★★★ SEAL of APPROVAL 1982 — Professional Software Programmers Association — Non Profit Tax Exempt 1982

NOT COPY PROTECTED

Complete System Capability Demo to Give You the Ability to Use the System Before You Purchase.

Versatile – *fits a wide variety of systems and applications . . .*

- Normal shift key operation for case shifting (optional).
- Can be used with or without lower case adapter.
- Supports optional Videx 80 column card (ideal for screen preview).
- Fast editor/formatter switching with RAM card installed.
- Supports Qume, NEC, Diablo, C. Itoh, Vista, Epson, many other printers.
- Uses standard text files – compatible with a wide range of other software, such as data base managers, user-written BASIC programs, and spelling dictionaries!

☐ Please send me a
WRITE AWAY DEMO DISK

Name _____
Company _____
Address _____
City _____
State/Zip _____

MSA **Midwest Software Associates**
P. O. Box 301
St. Ann, Missouri 63074

Midwest Software Associates • P. O. Box 301 • St. Ann, Missouri 63074
Phone: *Toll Free* 1-800-362-2421/Ext. 467 MISSOURI: 1-800-835-2246/Ext. 467

continued from page 44

When you either store (PUT) or retrieve (GET) a shape, you will be asked to enter its character code. There are two ways to enter this code:

1) By its ASCII value, a number from 0 to 255. It must be preceded by a "#".

2) By an ASCII symbol. You will then be asked to enter its type (flashing, inverse, control, normal or lowercase).

3) By pressing RETURN to select the default character. This default is displayed upon entering either the GET or PUT modes.

Methods 1 and 2 are really one and the same because the numbers 0-255 are the ASCII values that correspond to the Character Set (see page 15 of the *Apple Reference Manual*). This means that a shape with a code of 116 can be accessed by entering the characters 4 and then F (for flashing).

Once in the editor there are many commands available (see illustration #2). Motion keys are W for up, X for down, A for left, and S for right. Pressing the SPACE bar toggles the current pixel on/off (plot/don't plot).

Three continuous modes are available. Exit any of them by pressing either the SPACE bar or the ESC key.

1 "Continuous Plot" mode. Every time you move, you will plot a point.

2 "Continuous Erase" mode. Every time you move, the current pixel will be erased.

3 "Special Reverse" mode. Every time you move, you reverse the image. In other words, each pixel is turned "on" if previously "off" or "off" if previously "on".

The left arrow sets the high bit, and the right arrow clears it. This allows you to easily change colors. When either of these keys are pressed, you will be presented with the following choices:

Block, Line, Segment or All.

The "Block" option will only do the specified operation on the current character you are in. The "Line" option will do the entire horizontal line you are on. The "Segment" option will only do that line within the block. And "All" will do the entire shape.

ctrl I will inverse the shape (see above for options).

ctrl F will flip the high bit for each segment (see above for options).

" (shift 2 on an Apple) will allow you to select a new block size.

```
330 A =  VAL (MS$): IF A = 0 THEN
    A = H
340  PRINT A
350  IF A > 7 THEN  VTAB 5: PRINT
    8$"HORIZONTAL OUT OF RANGE (
    MAX=7)": VTAB 1: GOTO 310
360  IF A < 1 THEN  VTAB 5: PRINT
    8$"HORIZONTAL OUT OF RANGE M
    IN=1)": VTAB 1: GOTO 310
370  VTAB 5: PRINT  SPC( 31): PRINT
380  VTAB 3: PRINT "ENTER VERTICA
    L BLOCK SIZE ";: GET MS$
390  IF  ASC (MS$) = 27 AND TA <
     > 0 THEN 2330
400 X =  VAL (MS$): IF X = 0 THEN
    X = V
410  PRINT X
420  IF X > 4 THEN  VTAB 5: PRINT
    8$"VERTICAL OUT OF RANGE (MA
    X=4)": GOTO 380
430  IF X < 1 THEN  VTAB 5: PRINT
    8$"VERTICAL OUT OF RANGE (MI
    N=1)": GOTO 380
440  VTAB 5: PRINT  SPC( 31)
450  PRINT
460  PRINT "INVERSE BACKGROUND (Y
    /N)? ";: GET MS$: IF  ASC (M
    S$) = 27 AND TA <  > 0 THEN
    2330
470  IF MS$ = "Y" THEN I$ = "Y":I
     = 255:T$ = "0000"
480  PRINT I$
490  PRINT
500  PRINT : PRINT "SOUND (Y/N)?
    ";: GET MS$: IF  ASC (MS$) =
    27 AND TA <  > 0 THEN 2330\
510  IF  ASC (MS$) = 13 THEN MS$ =
    S$
520  IF MS$ = "N" THEN G$ = ""
530  IF MS$ <  > "N" THEN MS$ = "
    Y":8$ =  CHR$ (7)
540 S$ = MS$: PRINT MS$
550 H = A:V = X
560 0$ = "A":C$ = "N":0C$ = C$
570 0 =  ASC (0$) + 128
580  HGR
590 BY = V * H * 8
600  POKE 788,I: POKE ST,BY: CALL
    785
610  POKE 34,20
620  POKE 9,BY + 1
630  POKE 4,N5: POKE 5,N6: POKE 6
    ,1: POKE 7,1
640 A$(0) = "A":A$(1) = "H":A$(2)
    = "G"
650 A$(3) = "H"
660  FOR XX = 1 TO H
670 A$(0) = A$(0) + "BBBB":A$(1) =
    A$(1) + T$:A$(2) = A$(2) + "
    FFFF"
680 A$(3) = A$(3) + "IJKL"
690  NEXT XX
700 A$(0) = A$(0) + "C":A$(1) = A
    $(1) + "D":A$(2) = A$(2) + "
    E"
710 A$(3) = A$(3) + "D"
720  POKE 54,N3
730  POKE 55,N4: CALL 1002
740 TA =  INT ((31 - H * 4) / 2):
     HTAB TA
750  REM
XN=MIN. VALUE FOR XP, YN
    =MIN. VALUE FOR YP
760 XN = 8 + ((TA - 1) * 7):YN =
    9
770 XM = XN - 4 + 28 * H:YM = 5 +
    32 * V
780  REM
EDIT BOX
790 CX = 0:CY = 0
800 T2 = 0
810 XP = XN:YP = YN
820 T3 = 0
830  INVERSE
840  VTAB 1
850  PRINT A$(0)
860  FOR X = 1 TO V * 4
870  HTAB TA: PRINT A$(1): NEXT
880  HTAB TA: PRINT A$(2)
890  FOR Y = 1 TO V
900  FOR X = 1 TO H: VTAB Y: HTAB
    32 + X: PRINT "N": NEXT X
910  NEXT Y
920  VTAB 1: HTAB 33: POKE 2125,8
    1: PRINT "N": POKE 2125,145
930 OX = 0:OY = 1
940  POKE 2125,145
950  REM
RE-CONNECT HCG
960  POKE 54,N3: POKE 55,N4: CALL
    1002
970  REM
DISPLAY CHAR. AT SIDE OF
     SCREEN
980  POKE 4,N7: POKE 5,N8
990  POKE 6,H: POKE 7,V
1000  POKE  - 16368,0
1010  VTAB 9: HTAB 33
1020  INVERSE
1030  PRINT "@"
1040  IF OX = CX AND OY =  INT (C
    Y) + 1 THEN 1130
1050  POKE 6,1: POKE 7,1
1060  POKE 4,N5: POKE 5,N6
1070  POKE 2125,81: VTAB OY: HTAB
    33 + OX: PRINT "N"
1080  VTAB  INT (CY) + 1: HTAB CX
    + 33: PRINT "N"
1090 OX = CX:OY =  INT (CY) + 1
1100  POKE 2125,145
1110  POKE 6,H: POKE 7,V
1120  POKE 4,N7: POKE 5,N8
1130  XDRAW 1 AT XP,YP: XDRAW 2 AT
    XP,YP
1140  FOR X = 1 TO 10: NEXT
1150  XDRAW 1 AT XP,YP: XDRAW 2 AT
    XP,YP
1160 KEY =  PEEK ( - 16384): IF K
    EY <  = 127 THEN 1010
1170  POKE  - 16368,0
1180  IF T3 = 0 THEN 1210
1190  IF KEY = 215 OR KEY = 193 OR
    KEY = 196 OR KEY = 216 THEN
     GOSUB 3290
1200  REM
DOWN "X"
1210  IF KEY = 216 THEN 1570
1220  REM
UP "W"
1230  IF KEY = 215 THEN 1620
```

```
1240  REM
RIGHT "D"
1250  IF KEY = 196 THEN 1650
1260  REM
LEFT "A"
1270  IF KEY = 193 THEN 1700
1280  REM
PLOT/UNPLOT "SPACE"
1290  IF KEY = 160 THEN T3 = 0: GOTO
     1760
1300  REM
ERASE CHARACTER "!"
1310  IF KEY = 161 THEN 600
1320  REM
START ALL OVER "*"
1330  IF KEY = 162 THEN 240
1340  REM
STORE CHARACTER "P"
1350  IF KEY = 208 THEN 1850
1360  REM
GET CHARACTER "G"
1370  IF KEY = 199 THEN 2010
1380  IF KEY > 176 AND KEY < 180 THEN
     T3 = KEY - 176: GOTO 1010
1390  REM
ESC
1400  IF KEY = 155 THEN T3 = 0
1410  REM
DISK "CTRL D"
1420  IF KEY = 132 THEN 2190
1430  REM
ERASE CHARACTER SET ("@")
1440  IF KEY = 192 THEN 2370
1450  REM
CHECK FOR EXIT (CTRL X)
1460  IF KEY = 152 THEN  TEXT : HOME
     : NORMAL : PRINT D$"PR#0": END
     : REM CTRL X, END PROGRAM
1470  REM
CTRL U (SET HIGH BIT)
1480  IF KEY = 149 THEN  POKE NO,
     1: GOTO 2420
1490  REM
CTRL H (CLEAR HIGH BIT)
1500  IF KEY = 136 THEN  POKE NO,
     0: GOTO 2420
1510  REM
FLIP CHARACTER "?"
1520  IF KEY = 191 THEN 2610
1530  REM
CTRL I, INVERSE (EOR)
1540  IF KEY = 137 THEN 2630
1550  GOTO 1010
1560  REM

DOWN
1570 YP = YP + 4
1580 CY = CY + IV
1590  IF YP > YM THEN YP = YM:CY =
     0
```

G	GETs a character from the Character Set. The character may be entered as an ASCII value (*see HardCore Computing Update* 2.1, or page 15 of the *Apple II Reference Manual* for the proper ASCII codes). If an ASCII value is to be entered, precede it with the pounds sign (#) and press RETURN when done. If entered as a normal character, you will be asked to specify what type (flashing, control, lower case). RETURN defaults to Normal.
P	PUTs a character into the Character Set. This command works the same way as the GET command.
!	(shift "1" on an Apple) will cause the entire character to be erased.
ctrl D	catalogs the disk and asks whether you wish to SAVE or LOAD a Character Set. Press ESC to exit. When asked for the file name, RETURN will exit. Otherwise enter the file name and press RETURN.

ctrl @ clears the entire Character Set. When using this command, you will be asked to enter the entire word "YES" and press RETURN if you want to erase the Character Set. Every other entry exits this routine.

ctrl X exits from the editor.

NOTES: All the commands (except when using the #) require a single key press and do not require the pressing of RETURN (which is used to select defaults). To exit from any of the above options, press the ESC key (or RETURN if it is specified).

When Designing Letters

If you are designing letters, make sure you leave a pixel on both sides of the shape and one at the top and/or bottom so that your letters do not merge into each other.

Now that you have created a few characters and saved them, don't forget to save the entire Character Set. You are now ready to link Quick Draw to your own BASIC program.

As previously mentioned, each character has a number 0-255 associated with it. To use the HCG effectively, a fast method of passing the character number to the HCG had to be found. I decided to use the PRINT command. This means you can use VTAB, HTAB, and TAB to position your shape on the hi-res screen, then simply PRINT the ASCII character. For example, to PRINT the shape stored in character #0, you would first set the INVERSE mode and then PRINT "@", since 0 on the ASCII table (page 15 of the Reference Manual) is an inverse "@". For special characters, such as the Underline, you must set the proper mode (flashing, inverse or normal) and print the Applesoft ASCII (which can be found by subtracting 128 from the normal ASCII value). I have included a few examples of letters and how they would be called:

```
1600  GOTO 1010
1610  REM

UP
1620 YP = YP - 4:CY = CY - IV: IF
     YP < YM THEN YP = YM:CY = V -
     IV
1630  GOTO 1010
1640  REM

RIGHT
1650 XP = XP + 4
1660  IF XP > XM THEN XP = XM:CX =
     0:T2 = 0: GOTO 1010
1670 T2 = T2 + 1: IF T2 > 6 THEN
     T2 = 0:CX = CX + 1
1680  GOTO 1010
1690  REM

LEFT
1700 XP = XP - 4
1710  IF XP < XM THEN XP = XM:CX =
     H - 1:T2 = 6: GOTO 1010
1720 T2 = T2 - 1: IF T2 < 0 THEN
     T2 = 6:CX = CX - 1
1730  GOTO 1010
1740  REM

PLOT/UNPLOT
1750  REM
CLEAR COLLISION COUNT
1760  POKE 234,00
1770  XDRAW 1 AT XP,YP
1780 P = 1: IF  PEEK (234) = 0 THEN
     P =  - 1
1790  GOSUB 2640
1800 P =  PEEK (PO) + 2 ^ T2 * P
1810  IF P < 128 THEN  XDRAW 2 AT
     XP,YP
1820  POKE PO,P
1830  GOTO 1010
1840  REM

PUT CHARACTER
1850 MS$ = "SAVE AS"
1860  GOSUB 2870
1870  HOME : VTAB 21: PRINT "SAVE
      THIS CHARACTER UNDER THE FO
     LLOWING LETTER? ";
1880  IF  ASC (A$) < 32 THEN  PRINT
     "CTRL "; CHR$ ( ASC (A$) + 6
     4)" ";: GOTO 1930
1890  IF C$ = "C" THEN  PRINT "CT
     RL ";
1900  IF C$ = "L" THEN  PRINT "LW
     R ";
1910  IF C$ = "I" THEN  INVERSE
1920  IF C$ = "F" THEN  FLASH
1930  PRINT A$;: NORMAL : PRINT "
      (Y/N)? ";: GET B$: PRINT : IF
     B$ < > "Y" AND B$ < > CHR$
     (13) THEN  GOSUB 2900: GOTO
     1870
1940  HOME : VTAB 22: FLASH : HTAB
     11: PRINT "STORING CHARACTER
     ": FOR X = 0 TO 100: NEXT X
1950  GOSUB 2670
1960  POKE ST,BY
1970  CALL 796
1980  HOME
1990  GOTO 960
2000  REM

GET CHARACTER
2010 MS$ = "GET": GOSUB 2870
2020  GOSUB 2670
2030  POKE ST,BY
2040  CALL 807
2050 XP = XN:YP = YN:T2 = 0:T3 =
     0:CX = 0:CY = 0:SX = XP:SY =
     YP:NA = YM:NI = YN:BB = 00:E
     B = BY
2060  HOME
2070  VTAB 22: FLASH : HTAB (11):
      PRINT "GETTING CHARACTER"
2080  POKE 54,N3: POKE 55,N4
2090  CALL 1002
2100  VTAB 9: HTAB 33
2110  INVERSE
2120  PRINT "@"
2130  POKE 0,SX: POKE 1,SY: POKE
     2,NA: POKE 3,NI
2140  POKE ST,BB: POKE EN,EB
2150  CALL 5000
2160  PR# 0: HOME
2170  GOTO 960
2180  REM

ACCESS DISK
2190  GOSUB 3470: TEXT : HOME
2200  PRINT
2210  NORMAL : POKE  - 16368,0
2220  PRINT D$"CATALOG": GET A$: PRINT

2230  PRINT : PRINT
2240  VTAB 22: PRINT "LOAD OR SAV
     E CHARACTER SET (L/S)? ";: GET
     A$: PRINT
2250  IF  ASC (A$) = 27 THEN 2330

2260 L$ = ",A$E00"
2270  IF A$ = "L" THEN A$ = "BLOA
     D": GOTO 2300
2280  IF A$ = "S" THEN A$ = "BSAV
     E":L$ = L$ + ",L$800": GOTO
     2300
2290  PRINT G$"ILLEGAL ENTRY": GOTO
     2240
2300  PRINT "ENTER CHARACTER SET
     TO "A$;: INPUT " ";T1$
2310  IF  LEN (T1$) = 0 THEN 2330

2320  HOME : VTAB 12: PRINT A$"IN
     G "T1$: PRINT D$;A$;T1$;L$
2330  HOME : POKE 54,N3: POKE 55,
     N4: CALL 1002
2340  POKE  - 16304,0: POKE  - 16
     297,0
2350  GOTO 1010
2360  REM

ERASE CHARACTER SET
2370  NORMAL : PRINT D$"PR#0": INVERSE
     : VTAB 22: INPUT "ERASE THIS
      CHARACTER SET (YES/NO)? ";A
     $
2380  GOTO 960
2390  IF A$ = "YES" THEN  CALL 81
     0: GOTO 240
2400  GOTO 2330
2410  REM

CHANGE HIGH BIT
2420  GOSUB 3470: VTAB 23: PRINT
     G$G$;: IF  PEEK (NO) = 0 THEN
      PRINT "CLEAR";
```

SHAPE 23 (an inverse W):
set the INVERSE mode
and PRINT "W"

SHAPE 225 (a lower case A):
set the NORMAL mode
and PRINT CHR$(225-128)

SHAPE 210 (a normal R):
set the NORMAL mode
and PRINT "R"

Linking the HCG to BASIC

The first thing that must be done is to move the program above the hi-res page. Next LOAD the Character Set and Quick Draw. To do this, enter the following "header" as the first lines of your program:

10 IF PEEK (103) = 1 AND PEEK (104) = 64 THEN 30

20 POKE 103,1: POKE 104,64: POKE 16384,0: PRINT CHR$(4)"RUN program name here"

30 PRINT CHR$(4)"BLOAD QUICK DRAW"

40 PRINT CHR$(4)"BLOAD TABLES"

50 PRINT CHR$(4)"BLOAD name of character set,A$E00"

Line 10 checks to see if you are above the hi-res page. If not, it resets the pointers to load the program above the hi-res page and reruns the program (line 20). Line 30 loads Quick Draw into memory and line 40 loads in the appropriate Character Set.

The following pointers must be set to allow the HCG to operate correctly:

1) Zero page locations 4 and 5 contain the address of the Character Set we wish to use. This location can be changed if more than one Character Set is required. The normal location of the Character Set is 3584 decimal ($E00 hex).
POKE 4,0: POKE 5,14

2) Locations 6 and 7 contain the horizontal and vertical block sizes, respectively. Normally:
POKE 6,1: POKE 7,1

```
2430  IF  PEEK (HO) = 1 THEN  PRINT
    "SET";
2440  PRINT " HIGH BIT":CA = 844
2450  PRINT "BLOCK,ALL,LINE OR SE
    GMENT? ";: GET C$: HOME : PRINT
2460  IF C$ = "B" THEN 2560
2470  IF C$ = "A" THEN  POKE ST,0
    : POKE EN,BY: CALL CA:SX = X
    N:SY = YN:HA = YM:HI = YN:BB
     = 0:EB = BY: GOTO 2080
2480  IF C$ < > "S" THEN 2500
2490  GOSUB 2640:PO = PO - HG: POKE
    ST,PO: POKE EN,PO + 1: CALL
    CA:SX = XP - 4 * T2:SY = YP:
    BB = PO:EB = PO + 1:HI = YN:
    HA = YM: GOTO  HOME : GOTO 2
    080
2500  IF C$ < > "L" THEN 960
2510 T4 = CX: FOR X = 1 TO H:CX =
    X - 1: GOSUB 2640:PO = PO -
    HG
2520  POKE ST,PO: POKE EN,PO + 1:
     CALL CA: POKE 0,XN + 28 + (
    X - 1): POKE 1,YP: POKE 2,YN
    : POKE 3,YN
2530  POKE ST,PO: POKE EN,PO + 1:
     CALL 5888
2540  NEXT X:CX = T4
2550  GOTO 960
2560 T4 = CY:CY =  INT (CY): GOSUB
    2640:CY = T4:PO = PO - HG
2570  POKE ST,PO: POKE EN,PO + 8:
     CALL CA
2580 SX = XP - 4 * T2:SY = YP - 4
     *  INT (8 * (CY -  INT (CY)
    )):HA = YM:HI = YN:BB = PO:E
    B = PO + 8
2590  HOME : GOTO 2080
2600  REM

FLIP CHARACTER
2610  GOSUB 3470: VTAB 23: PRINT
    G$G$"FLIP CHARACTER":CA = 88
    4: GOTO 2450
2620  REM

INVERSE (EOR)
2630  GOSUB 3470: VTAB 23: PRINT
    G$G$"INVERSE CHARACTER":CA =
    868: GOTO 2450
2640  REM

CALC. WHERE BYTE IS
2650 PO = HG + ( INT (CY) + CX *
    V) * 8 + (CY -  INT (CY)) *
    8
2660  RETURN
2670  REM

UPDATE DEFAULT CHAR.
2680 O$ = A$
2690 A = 0
2700  IF  LEN (A$) = 0 THEN 2850
2710 A =  ASC (A$) + 128
2720  IF C$ = "" THEN 2850
2730  IF  LEN (C$) = 0 OR C$ = "N
    " OR  ASC (C$) = 13 THEN 285
    0
2740  IF C$ < > "L" AND C$ < >
    "N" AND C$ < > "I" AND C$ <
    > "C" AND C$ < > "F" THEN
    C$ = "": GOTO 2850
2750  IF C$ < > "C" THEN 2780
2760  IF A > 191 AND A < 224 THEN
    A = A - 64
2770  GOTO 2850
2780  IF C$ < > "L" THEN 2800
2790  IF A > 191 AND A < 224 THEN
    A = A + 32: GOTO 2850
2800  IF C$ = "F" THEN 2840
2810 A = A - 128
2820  IF A > 63 THEN A = A - 64
2830  GOTO 2850
2840 A = A - 64: IF A > 127 THEN
    A = A - 64
2850 OC$ = C$:O = A:X = LO + A *
    8 - 1: POKE 27, INT (X / 256
    ): POKE 26,X -  INT (X / 256
    ) * 256
2860  RETURN
2870  REM

GET SAVE/LOAD CHAR.
2880 A =  - 1
2890  GOSUB 3470
2900  PRINT G$G$
2910  HOME : VTAB 21: PRINT MS$"
    WHAT CHARACTER ($"O") ";
2920  IF  LEN (O$) = 0 THEN 3000
2930  IF OC$ = "F" THEN  FLASH
2940  IF OC$ = "I" THEN  INVERSE

2950  IF  ASC (O$) < 32 THEN  PRINT
    "CTRL " CHR$ ( ASC (O$) + 64
    );: GOTO 2990
2960  IF OC$ = "C" THEN  PRINT "C
    TRL ";
2970  IF OC$ = "L" THEN  PRINT "L
    HR ";
2980  PRINT O$;
2990 C$ = OC$
3000  NORMAL : PRINT " ? ";: GET
    A$
3010  IF  ASC (A$) = 13 THEN A$ =
    O$: RETURN
3020  IF A$ = "" OR  LEN (A$) < 1
     THEN A$ = O$: RETURN
3030  IF  ASC (A$) = 27 THEN  POP
    : HOME : POKE 54,N3: POKE 55
    ,N4: CALL 1002: GOTO 1010
3040  PRINT A$;
3050  IF A$ = "#" THEN  INPUT "";
    B$:A$ = A$ + B$
3060  IF  LEFT$ (A$,1) = "#" AND
     LEN (A$) < > 1 THEN 3170
3070  IF  ASC (A$) < 20 THEN C$ =
    "N": RETURN
3080  IF  ASC (A$) > 63 THEN 3120

3090  PRINT : PRINT "WHAT TYPE NO
    R/INV/FLS? ";: GET C$: PRINT
    : IF  ASC (C$) = 27 THEN 313
    0
3100  IF C$ < > "N" AND C$ < >
    "I" AND C$ < > "F" AND C$ <
    > CHR$ (13) THEN  PRINT G$
    "ILLEGAL ENTRY": GOTO 3090
3110  RETURN
3120  PRINT : PRINT "WHAT TYPE NO
    R/INV/FLS/CTR/LOW? ";: GET C
    $: PRINT
3130  IF  ASC (C$) = 27 THEN  POP
    : HOME : POKE 54,N3: POKE 55
    ,N4: CALL 1002: GOTO 1010
3140  IF C$ < > "N" AND C$ < >
    "I" AND C$ < > "F" AND C$ <
     > "C" AND C$ < > "L" AND C
    $ < > CHR$ (13) THEN  PRINT
    G$"ILLEGAL ENTRY": GOTO 3090
```

To link the HCG, we need to force the PRINT command to go to the HCG instead of to the normal routine for printing characters. To do this, change $37 and $38 (the print output hooks) to point to $803 and do a call to DOS to let it know what's going on:

POKE 54,3: POKE 55,8: CALL 1002

Now the HCG is linked and ready to go. To disable the generator,

PRINT CHR$(4)"PR#0"

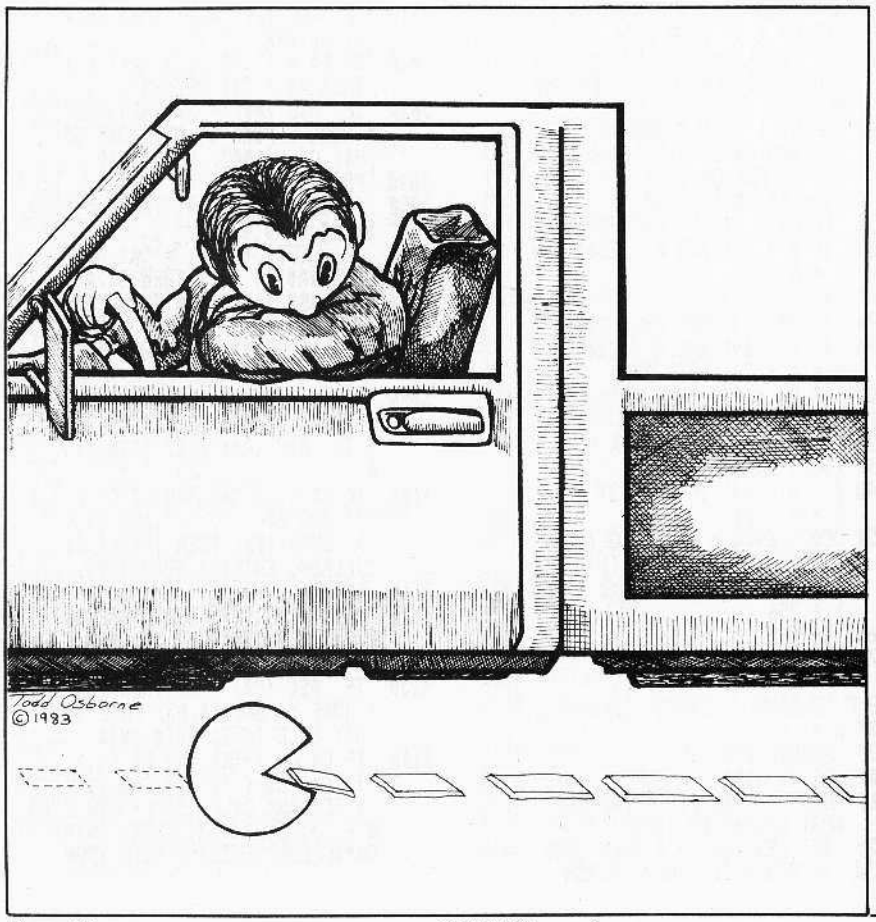
While the generator is linked, all characters will be printed to the hi-res screen, even error messages (so be careful or your error message may be a series of strange shapes and symbols).

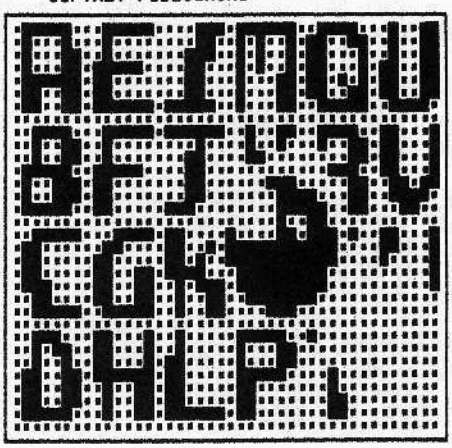
To do all of the above, add these lines to the header you started:

60 POKE 4,0: POKE 5,14
70 POKE 6,1: POKE 7,1
80 POKE 54,3: POKE 55,8: CALL 1002

To discover just how easy it is to use, play with the demo for Quick Draw (Orde's Space Raid). After playing around with it for a while, return here for some additional features of Quick Draw.

```
3150  IF  ASC (A$) + H * V - 1 >
      95 AND C$ = "L" THEN  FLASH
      : HTAB 12: PRINT G$"NOT ENOU
      GH ROOM": FOR X = 1 TO 800: NEXT
      : NORMAL : GOTO 2910
3160  RETURN
3170 A$ =  RIGHT$ (A$, LEN (A$) -
      1)
3180 A =  VAL (A$): IF  STR$ (A) <
       > A$ OR A < 0 OR A > 255 THEN
       PRINT : PRINT G$"ILLEGAL EN
      TRY": GOTO 2910
3190  IF A + H * V - 1 > 255 THEN
      FLASH : HTAB 12: PRINT G$"N
      OT ENOUGH ROOM": FOR X = 1 TO
      800: NEXT : NORMAL : GOTO 29
      10
3200  IF A < 32 THEN A$ =  CHR$ (
      A + 192):C$ = "I": GOTO 3270

3210  IF A < 63 THEN A$ =  CHR$ (
      A + 128):C$ = "I": GOTO 3270

3220  IF A < 96 THEN A$ =  CHR$ (
      A + 128):C$ = "F": GOTO 3270

3230  IF A < 128 THEN A$ =  CHR$
      (A + 64):C$ = "F": GOTO 3270

3240  IF A < 160 THEN A$ =  CHR$
      (A + 64):C$ = "C": GOTO 3270

3250  IF A < 224 THEN A$ =  CHR$
      (A):C$ = "N": GOTO 3270
3260  IF A > 223 THEN A$ =  CHR$
      (A - 32):C$ = "L"
3270 A$ =  CHR$ ( ASC (A$) - 128)
      :D$ = A$
3280  RETURN
3290  REM

CHECK FOR PLOTTING
3300  POKE 234,0
3310  IF T3 <  > 1 THEN 3360
3320  DRAW 1 AT XP,YP: IF  PEEK (
      234) <  > 0 THEN  RETURN
3330  GOSUB 2640: POKE P0, PEEK (
      P0) + 2 ^ T2
3340  IF  PEEK (P0) < 128 THEN  XDRAW
      2 AT XP,YP
3350  RETURN
3360  IF T3 <  > 2 THEN 3420
3370  HCOLOR= 0: DRAW 1 AT XP,YP:
      HCOLOR= 3
3380  IF  PEEK (234) <  > 0 THEN
      RETURN
3390  GOSUB 2640: POKE P0, PEEK (
      P0) - 2 ^ T2
3400  IF  PEEK (P0) < 128 THEN  XDRAW
      2 AT XP,YP
3410  RETURN
3420  POKE 234,0
3430  XDRAW 1 AT XP,YP:P = 1: IF
      PEEK (234) = 0 THEN P =  -
      1
3440  GOSUB 2640: POKE (P0), PEEK
      (P0) + 2 ^ T2 * P
3450  IF  PEEK (P0) < 128 THEN  XDRAW
      2 AT XP,YP
3460  RETURN
3470  NORMAL : PRINT D$"PR#0": PRINT
3480  RETURN
3490 X =  PEEK (218) +  PEEK (219
      ) * 256:Y =  PEEK (222)
3500  STOP
3510  REM

WRITTEN BY


3520  REM
ROBB CANFIELD

3530  REM
COPYRIGHT 1982

3540  REM

SOFTKEY PUBLISHING
```





QD.Editor.Util.Obj *(Step Three)*

```
1000 *--------------------------------
1010 *                               *
1020 *    ROUTINES FOR THE EDITOR    *
1030 *                               *
1040 *       BY ROBB CANFIELD        *
1050 *                               *
1060 *        COPYRIGHT 1982         *
1070 *                               *
1080 *      SOFTKEY PUBLISHING       *
1090 *                               *
1100 *--------------------------------
1110
1120
1130 START    .EQ $0B       THE LINK BETWEEN BASIC AND MACHINE
1140 END      .EQ $0C       ENDING BYTE
1150 MODE     .EQ $19
1160 CHAR.SET .EQ $E00      CHARACTER SET CURRENTLY IN MEMORY
1170 WORK.CHAR .EQ $1800-1 AREA FOR THE CHARACTER WE ARE WORKING ON
1180
1190 POINTER  .EQ $1A,$1B  POINTERS FOR VARIOUS MOVE ROUTINES
1200
1210
1220
1230          .OR $311 .   PUT RIGHT BELOW THE SHAPE TABLE
1240          .TF QD.EDITOR.UTIL.OBJ
1250
1260
1270
1280
1290 *--------------------------------
1300 * ERASE.MEM WILL ERASE A        *
1310 * DESIGNATED NUMBER OF BYTES    *
1320 * IN MEMORY STARTING AT $4000.  *
1330 * THE NUMBER OF BYTES TO ERASE  *
1340 * -1 ARE STORED IN $D0 (DEC.    *
1350 * 208).                         *
1360 *                               *
1370 * THIS ROUTINE STARTS AT $311   *
1380 * OR 785 DEC.                   *
1390 *--------------------------------
1400
1410
1420 ERASE.MEM LDY START   GET NUMBER OF BYTES TO ERASE
1430           LDA #$00    PUT $00 INTO THESE BYTES
1440 CONT.ERASE STA WORK.CHAR,Y ERASE WHATEVER CHARACTER WAS HERE
1450           DEY         DO NEXT BYTE
1460           BNE CONT.ERASE IF NOT DONE THEN ERASE SOME MORE
1470           RTS         DONE SO RETURN TO BASIC
1480
1490
1500
1510 *--------------------------------
1520 * MOVE.TO: MOVES A CHARACTER    *
1530 * FROM THE WORK BUFFER TO THE   *
1540 * CHARACTER SET. THE NUMBER OF  *
1550 * BYTES TO MOVE IS STORED AT    *
1560 * $D0 (DEC. 208) AND THE        *
1570 * LOCATION TO MOVE THEM TO (-1) *
1580 * IS STORED AT $D1, $D2 (LOW,   *
1590 * HIGH) (DEC. 209, 210)         *
1600 *                               *
1610 * THIS ROUTINE STARTS AT $31C   *
1620 * OR 796 DEC.                   *
1630 *--------------------------------
1640
1650
1660 MOVE.TO  LDY START    GET NUMBER OF BYTES TO MOVE
1670 CONT.MOVE.TO LDA WORK.CHAR,Y GET BYTE TO MOVE
```

Advanced Use of Quick Draw

Since you are reading this, I assume you have become fairly familiar with the HCG. As you probably noticed from the demo, all shapes that were drawn had to be erased by drawing a blank over them. There is another way. Quick Draw is set up to allow exclusive "OR" (EOR) drawing to be done, as well as "Normal". If the EOR mode is set, the shape may be drawn without overwriting any background scenery that you may have drawn (much like XDRAWing with shape tables). The shape is then erased by drawing it again in the same spot. You set the mode for drawing by poking a certain location in the HCG with one of 2 values.

To set the STORE mode (normal):
POKE 2125,145
or from machine
$84D:9D

To set the EXCLUSIVE OR mode
POKE 2125,81
or from machine
$84D:51

There are many more possibilities yet to be discovered in using Quick Draw. Let your imagination run free and see what you can create.

How Does It Work?

A hi-res character generator is really very simple. Instead of calculating the vector, you simply grab a byte from one location and put it in another. Looking at it in this way, a HCG is nothing more than an elaborate Memory Move routine. To get a true idea of how this works, I have included the fully remarked source code. Even if you are not a machine language programmer, the remarks will let you know what is BASICly going on.

The editor is a bit more complicated. It has many machine language links that speed up the editing of shapes tremendously and, as such, is very hard to explain. I have highlighted each routine with a REM. One machine language routine of special interest is GET. This machine language subroutine enlarges the current shape and places it in the edit screen. Even though this routine is in machine language, it is still slow (you should have seen it in BASIC).

Here is a prime example of how shape tables and a HCG complement each other: the cursor is a vector shape. Quick Draw was not developed to replace shape tables, only to speed up certain types of graphics. Shape tables are still very valuable when speed is not of the essence and maneuverability with minimum code is.

Locations of Interest

The following is a list of the machine language routines used by the editor. The source code explains how to use each routine from machine language or BASIC.

GET: 5888 or $1700
ERASE: 818 or $332
FLIP: 884 or $374
EDITOR CHARACTER SET: 3072 or $D00
WORK BUFFER: 6144 or $1800
MOVE FROM BUFFER: 807 or $327
MOVE TO BUFFER: 796 or $31C
CHANGE HIGH BIT AND INVERSE: 844 or $34C

Other Notes

To change the characters I use for the editor, LOAD the file EDITOR SET. I only use the inverse characters. All block sizes are 1 by 1.

```
1680          STA (POINTER),Y STORE BYTE IN CHARACTER SET
1690          DEY
1700          BNE CONT.MOVE.TO IF NOT DONE CONTINUE
1710          RTS             DONE, SO RETURN TO BASIC
1720
1730
1740 *--------------------------------
1750 *  MOVE.FROM: MOVES A CHARACTER  *
1760 * FROM THE CHARACTER SET TO THE  *
1770 * EDITOR'S WORK BUFFER. NUMBER   *
1780 * BYTES TO MOVE IN $D0 (DEC.     *
1790 * 208) AND CHARACTER LOCATION    *
1800 * (FROM SPOT) -1, IN $D1, $D2    *
1810 * (DEC. 209,210), THEN JSR $327  *
1820 * OR CALL 807.                   *
1830 *--------------------------------
1840
1850
1860 MOVE.FROM LDY START   GET NUMBER OF BYTES TO MOVE
1870 CONT.MOVE.FROM
1880          LDA (POINTER),Y GET BYTE TO MOVE
1890          STA WORK.CHAR,Y STORE BYTE IN WORK BUFFER
1900          DEY
1910          BNE CONT.MOVE.FROM
1920          RTS
1930
1940
1950 *--------------------------------
1960 * CLEARS CHARACTER SET IN        *
1970 * MEMORY. TO USE "CALL 818".     *
1980 * MONITOR $332.                  *
1990 *--------------------------------
2000
2010          LDA /CHAR.SET POINT TO CHARACTER SET
2020          STA STORE+2
2030          LDA #CHAR.SET
2040
2050 ERASE    LDX $$8       NUMBER OF PAGES TO ERASE
2060 ERASE2   LDY $$0       EACH PAGE HAS 256 BYTES
2070 STORE    STA CHAR.SET ERASE
2080          DEY           NEXT BYTE
2090          BNE STORE     CONTINUE WITH BYTES
2100          DEX           NEXT PAGE
2110          BEQ DONE.ERASE FINISHED?
2120          INC STORE+2   GET NEXT PAGE
2130          BPL ERASE2    FORCED JUMP
2140 DONE.ERASE RTS         DONE. SO RETURN
2150
2160
2170 *--------------------------------*
2180 * SET HIGH BIT IN WORK BUFFER    *
2190 * MONITOR 34C, CALL 844          *
2200 * BYTE TO START ON- $D1 (209)    *
2210 * BYTE TO STOP ON - $D2 (210)    *
2220 * MODE 0-CLEAR, 1-SET -$D3 (211)*
2230 *--------------------------------*
2240
2250
2260 SET.HIGH.BIT
2270          LDY START     START ON BYTE
2280 HIGH.BIT LDA WORK.CHAR+1,Y GET BYTE TO CONVERT
2290          LDX MODE      CHECK MODE
2300          BEQ CLEAR
2310          ORA #$80      SET HIGH BIT
2320          BMI SET.BIT
2330 CLEAR    AND #$7F      CLEAR HIGH BIT
2340 SET.BIT  STA WORK.CHAR+1,Y SAVE CONVERTED BYTE
2350          INY
2360          CPY END       DONE?
2370          BNE HIGH.BIT NO
2380          RTS
2390
2400
2410 *--------------------------------
2420 * EXCLUSIVELY OR BYTES (INVERSE) *
2430 * MONITOR 364, CALL 868          *
2440 * STARTING BYTE-ST               *
2450 * ENDING BYTE-EN                 *
2460 *--------------------------------
2470
2480
2490 EOR      LDY START     GET FIRST BYTE
2500 .1       LDA WORK.CHAR+1,Y GET BYTE
2510          EOR #$7F      INVERSE IT
2520          STA WORK.CHAR+1,Y SAVE IT
2530          INY
2540          CPY END       DONE?
2550          BNE .1
2560          RTS
2570
2580
2590 *--------------------------------
2600 * FLIP CURRENT HIGH BIT          *
2610 * MONITOR $374, CALL 884         *
2620 * BEGIN AT BYTE ST               *
2630 * END AT BYTE EN                 *
2640 *--------------------------------
2650
2660
2670 FLIP     LDY START     GET BEGINNING BYTE
2680 FLIP.CONT LDA WORK.CHAR+1,Y GET BYTE
2690          BMI CLEAR.BIT
2700          ORA #$80      SET HIGH BIT
2710          BMI NEXT
2720 CLEAR.BIT AND #$7F     CLEAR HIGH BIT
2730 NEXT     STA WORK.CHAR+1,Y SAVE CONVERTED BYTE
2740          INY           GET NEXT BYTE
2750          CPY END       DONE?
2760          BNE FLIP.CONT NO. SO CONTINUE
2770          RTS
```

Get.Obj *(Step Four)*

```
1000 *--------------------------------
1010 *      THE GET ROUTINE
1020 *
1030 *      BY ROBB CANFIELD
1040 *    COPYRIGHT BY SOFTKEY
1050 *--------------------------------
1060
1070
1080 XP       .EQ $00
1090 YP       .EQ $01
1100 YN       .EQ $02
1110 YN       .EQ $03
1120 SHAPE1   .EQ $3D6
1130 SHAPE2   .EQ $3DE
1140 DRAW     .EQ $F601
1150 SETHCOL  .EQ $F6EC
1160 HPOSN    .EQ $F411
1170 TEMP     .EQ $8
1180 XSAVE    .EQ $9
1190 YSAVE    .EQ $A
1200 START    .EQ $B        START BYTE
1210 END      .EQ $C        ENDING BYTE
1220 BIT      .EQ $FD       BIT TO DRAW, >=0 MEANS DRAW ALL
1230 XP2      .EQ $FE       HOLDER FOR X-POSITION
1240 WORK.BUFFER .EQ $1800
1250
1260
1270          .OR $1700
1280          .TF GET.OBJ
1290
1300 GET      LDY START     GET WHERE TO START
1310          LDX XP        SAVE X-POSITION
1320          STX XP2
1330 NEXT.BYTE LDA WORK.BUFFER,Y
1340          PHP           SAVE STATUS
1350          AND #$7F      IGNORE COLOR BIT
1360          LDX BIT       GET BIT NUMBER
1370          CPX #$8       SKIP SEPARATE BIT ROUTINE
1380          BCS .1        YES
1390          DEX
1400 .2       LSR           GET PROPER BIT
1410          DEX
1420          BPL .2
1430          LDX XP2       GET ORIGINAL X-POSITION
1440          STX XP
1450          LDX #$1       SET COUNTER
1460          BNE BIT.LOOP FORCED BRANCH
1470 .1       LDX #$7       LOOP FOR BITS
1480 BIT.LOOP JSR SAVEALL
1490          LSR
1500          STA TEMP
1510          BCS DRAW.OUTSIDE
1520          LDX #$00      HCOLOR=0
1530          JSR SET.POSN
1540          LDY /SHAPE1
1550          LDX #SHAPE1
1560          LDA #$0
1570          JSR DRAW
1580          JMP SET.CENTER+1
1590 DRAW.OUTSIDE LDX #$03 HCOLOR=3
1600          JSR SET.POSN POSITION SHAPE TABLES
1610          LDY /SHAPE1   XDRAW 1
1620          LDX #SHAPE1
1630          LDA #$0       ZERO ROTATION
1640          JSR DRAW      XDRAW SHAPE
1650          PLP           GET STATUS BACK
1660          PHP
1670          BMI SET.CENTER+1 CLEAR/SET CENTER SQUARE
1680          LDX #$00      CLEAR CENTER
1690 SET.CENTER BIT $03A2  LDX #$03 HIDDEN
1700          JSR SET.POSN
1710          LDY /SHAPE2   XDRAW 2
1720          LDX #SHAPE2
1730          LDA #$00
1740          JSR DRAW
1750 NEXT.BIT JSR RESTORE
1760          CLC           CLEAR OLD POINT
1770          LDA XP        INCREMENT POSITION TO PLOT
1780          ADC #$4
1790          STA XP
1800          LDA TEMP
1810          DEX
1820          BNE BIT.LOOP
1830 LINE     PLP
1840          CLC
1850          LDA YP
1860          ADC #$4
1870          CMP YN
1880          BCC NEXT.LINE
1890          BEQ NEXT.LINE
1900 NEXT.COLUMN LDX YN
1910          STX YP
1920 GET.NEXT.BYTE INY
1930          CPY END       DONE?
1940          BNE NEXT.BYTE
1950 EXIT     RTS
```

continued on page 64

Text Graphics *continued from page 13*

The following routines are just a few of the more imaginative graphic displays that use only the text page.

```
0 REM BOX 1
10 TEXT : HOME : COLOR= 10: FOR I = 0 TO 18 STEP 2
19 REM (USE LO-RES ON TEXT PAGE)
20 VLIN 20 - I,20 + I AT 20 + I: VLIN 20 - I,20 + I AT
    20 - I
30 HLIN 20 - I,20 + I AT 20 - I: HLIN 20 - I,20 + I AT
    20 + I
39 REM (ALTER TEXT WINDOW AND ERASE)
40 NEXT : POKE 32,3: POKE 33,34: POKE 34,2: POKE 35,
    19: HOME
50 VTAB 19: HTAB 9: PRINT "YOUR MENU GOES HERE"
59 REM (SCROLL THE MESSAGE UP)
60 FOR A = 1 TO 15: PRINT : FOR I = 1 TO 50: NEXT : NEXT
    : GOTO 50
```



```
0 REM BOX 2
10 TEXT : HOME : COLOR= 10
20 FOR I = 2 TO 18 STEP 2: VLIN 20 - I,20 + I AT 20 +
    I: VLIN 20 - I,20 + I AT 20 - I
30 HLIN 20 - I,20 + I AT 20 - I: HLIN 20 - I,20 + I AT
    20 + I
40 POKE 32,22 - I: POKE 33,(I - 1) * 2: POKE 34,11 -
    I / 2: POKE 35,10 + I / 2
50 HOME : FOR J = 1 TO 50: NEXT : NEXT
60 A$ = "YOUR MENU GOES HERE": FOR A =  LEN (A$) TO 3
    0:A$ = A$ + "-": NEXT
69 REM (SCROLL MESSAGE TO THE LEFT)
70 A$ =  RIGHT$ (A$, LEN (A$) - 1) +  LEFT$ (A$,1)
80 VTAB 10: HTAB 2: PRINT A$: FOR A = 1 TO 50: NEXT
    : GOTO 70
```

```
0 REM BOX AWAY!
10 HOME : COLOR= 10: VTAB 10: HTAB 8: PRINT "AFTER Y
    OU CHOOSE..."
19 REM (SHRINKING SQUARES)
20 FOR J = 1 TO 1000: NEXT : FOR I = 18 TO 2 STEP  -
    2: HOME
30 VLIN 20 - I,20 + I AT 20 + I: VLIN 20 - I,20 + I AT
    20 - I
40 HLIN 20 - I,20 + I AT 20 - I: HLIN 20 - I,20 + I AT
    20 + I
50 FOR J = 1 TO 50: NEXT : NEXT : HOME
60 VTAB 10: PRINT "THE MENU VANISHES... BY SHRINKING
    AWAY."
```

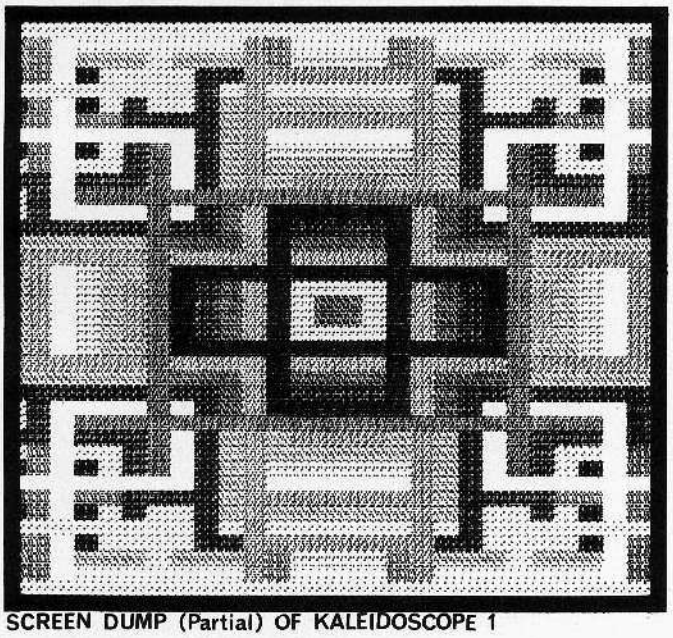
```
0 REM TEXT MARQUEE 1
5 GOTO 90
10 VTAB 1: HTAB 11 - Q: PRINT C$;: HTAB 21 - Q: PRINT
    C$;: HTAB 31 - Q: PRINT C$;: HTAB 41 - Q: PRINT
    C$;
20 VTAB Q: HTAB 1: PRINT C$
22 VTAB 10 + Q: HTAB 1: PRINT C$
30 VTAB 20: HTAB Q: PRINT C$;: HTAB 10 + Q: PRINT C$
    ;: HTAB 20 + Q: PRINT C$;: HTAB 30 + Q: PRINT C$
    ;
40 VTAB 21 - Q: HTAB 40: PRINT C$
42 VTAB 11 - Q: HTAB 40: PRINT C$: RETURN
50 IF  PEEK ( - 16384) < 128 THEN  RETURN
60 POKE  - 16368,0: IF M = 1 THEN M = 0: INVERSE : RETURN
70 M = 1: NORMAL : RETURN
90 HOME :A$ = "*":B$ = " ": VTAB 10: HTAB 12: PRINT
    "THE CORE MARQUEE": VTAB 12: HTAB 12: PRINT "PRE
    SS ANY KEY..."
100 FOR Q = 1 TO 10
199 REM LIGHTS OFF!
200 C$ = B$: GOSUB 10: GOSUB 50
299 REM LIGHTS BACK ON!
300 C$ = A$: GOSUB 10: NEXT : GOTO 100
```

Lo-Res Graphics *continued from page 18*

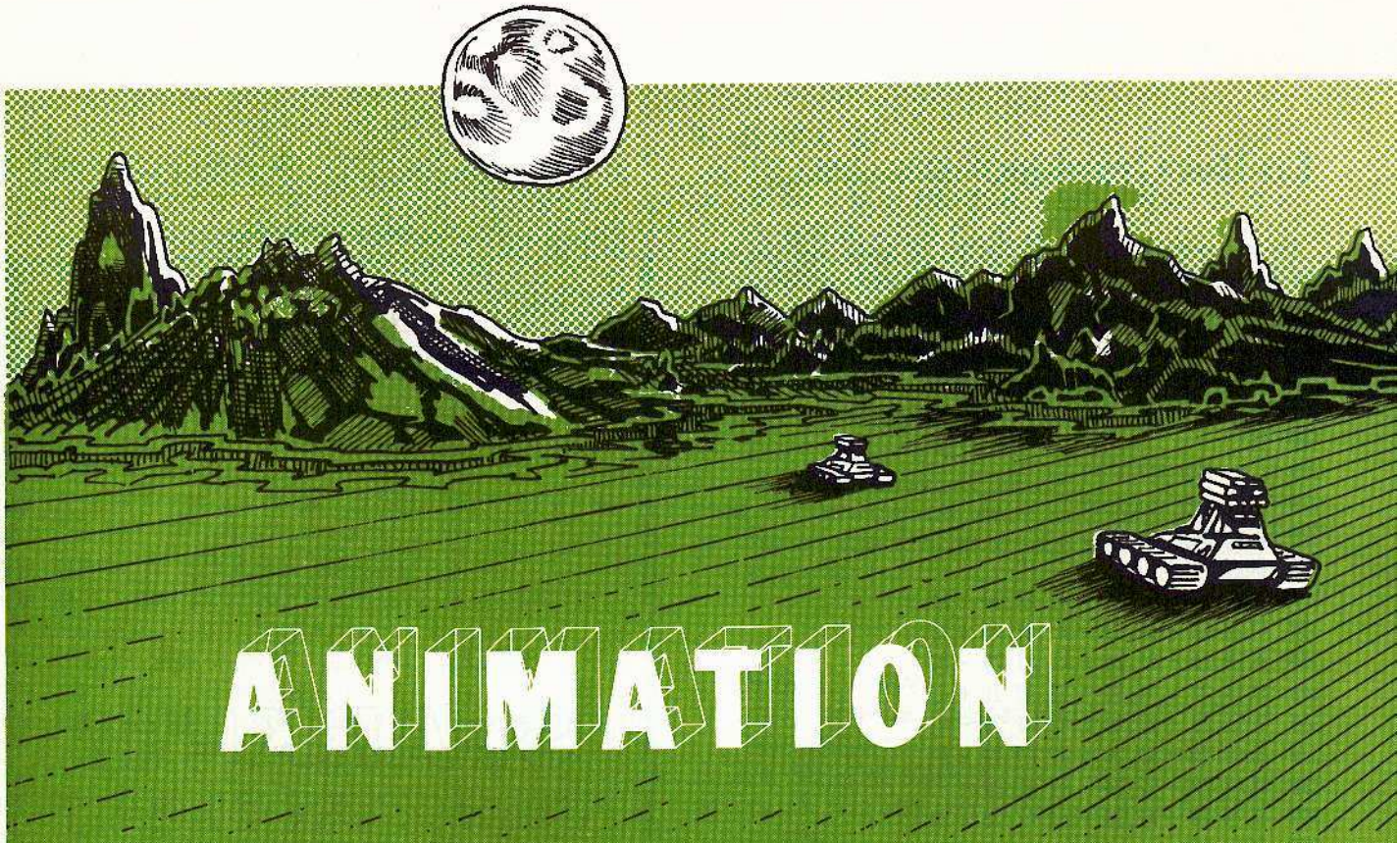
Lo-res color is simple and useful for really colorful displays. Programs directed toward children often use the lo-res screen, but some very interesting arcade games also use lo-res.

```
0  REM LO-RES KALEIDOSCOPE 1
100  GR
200 R1 =  RND (1) * 19
210 R2 =  RND (1) * 19
220 RC =  RND (1) * 16
300  COLOR= RC
310  VLIN 20 - R1,20 + R1 AT 20 - R2
320  VLIN 20 - R1,20 + R1 AT 20 + R2
330  HLIN 20 - R1,20 + R1 AT 20 - R2
340  HLIN 20 - R1,20 + R1 AT 20 + R2
360  VLIN 20 - R2,20 + R2 AT 20 - R1
370  VLIN 20 - R2,20 + R2 AT 20 + R1
380  HLIN 20 - R2,20 + R2 AT 20 - R1
390  HLIN 20 - R2,20 + R2 AT 20 + R1
990  GOTO 200
```

```
0  REM LO-RES KALEIDOSCOPE 2
10  GR
20  FOR A = 3 TO 50: FOR B = 1 TO 19: FOR C = 0 TO 19

30 D = B + C: COLOR= C * 3 / (B + 3) + B * A / 12
40  PLOT B,D: PLOT D,B: PLOT 40 - B,40 - D: PLOT 40 -
    D,40 - B: PLOT D,40 - B: PLOT 40 - B,D: PLOT B,4
    0 - D: PLOT 40 - D,B
50  NEXT : NEXT : NEXT : GOTO 20
```



SCREEN DUMP (Partial) OF KALEIDOSCOPE 1



ANIMATION

Of all the different aspects of Apple graphics that have been shown (memory, text, lo-res, hi-res, color, vectors, and block graphics), the real test of a program that uses graphics can be summed up in a single word:

Motion.

Depending on the object that must be moved, certain forms of animation are more appropriate. For example, a bullet (or missile) travels quickly, so it is not necessary to provide bit-by-bit motion. Using a character generator and printing consecutive positions that are seven or eight pixels away or farther is sufficient. For a slug that travels slowly, consecutive images must be only a couple of bits apart.

A problem novice animators discover is that color is difficult to maintain unless certain precautions are observed. White and black are easy to maintain, but other colors can only be preserved when consideration is given to that old odd/even pixel dilemma.

One solution is to limit horizontal motion to multiples of two characters. This allows a shape to remain on either even or odd bytes, but unless the animation is very, very fast, it will appear "jumpy" and uneven. Another solution is available to animators using a character generator. In this case, at least two shapes must be created in order to preserve color: one for even bytes (character positions) and one for odd bytes.

The best solution is to move shapes by two pixels horizontally. It's easy with a shape table. But it's more difficult when using a block (character) generator because several overlapping shapes must be created, each two pixels offset from the previous shape. The illustration "Move By Seven Shapes" shows that a block is only seven pixels across, so a given two-pixel pattern is repeated every two blocks (14 pixels), and it takes seven shapes (six of them spanning two whole characters each) to move to the next position where the pattern begins to repeat itself. That means that seven shapes must be created, using 13 character-blocks, to accomplish the horizontal, two-pixel movement of a single shape.

BASIC programs have the most problems with animation, because of the relatively slow speed of Applesoft. One solution would be to compile the Applesoft program. Another solution is to use hi-res machine routines by CALLing them from BASIC. An example of one such routine is the *Quick Draw* program in this issue.

The following program is an example of animation using *Quick Draw*. It's called *Space Raid*, by Rich Orde. Originally submitted as a text game, it was elevated to hi-res using *Quick Draw*.

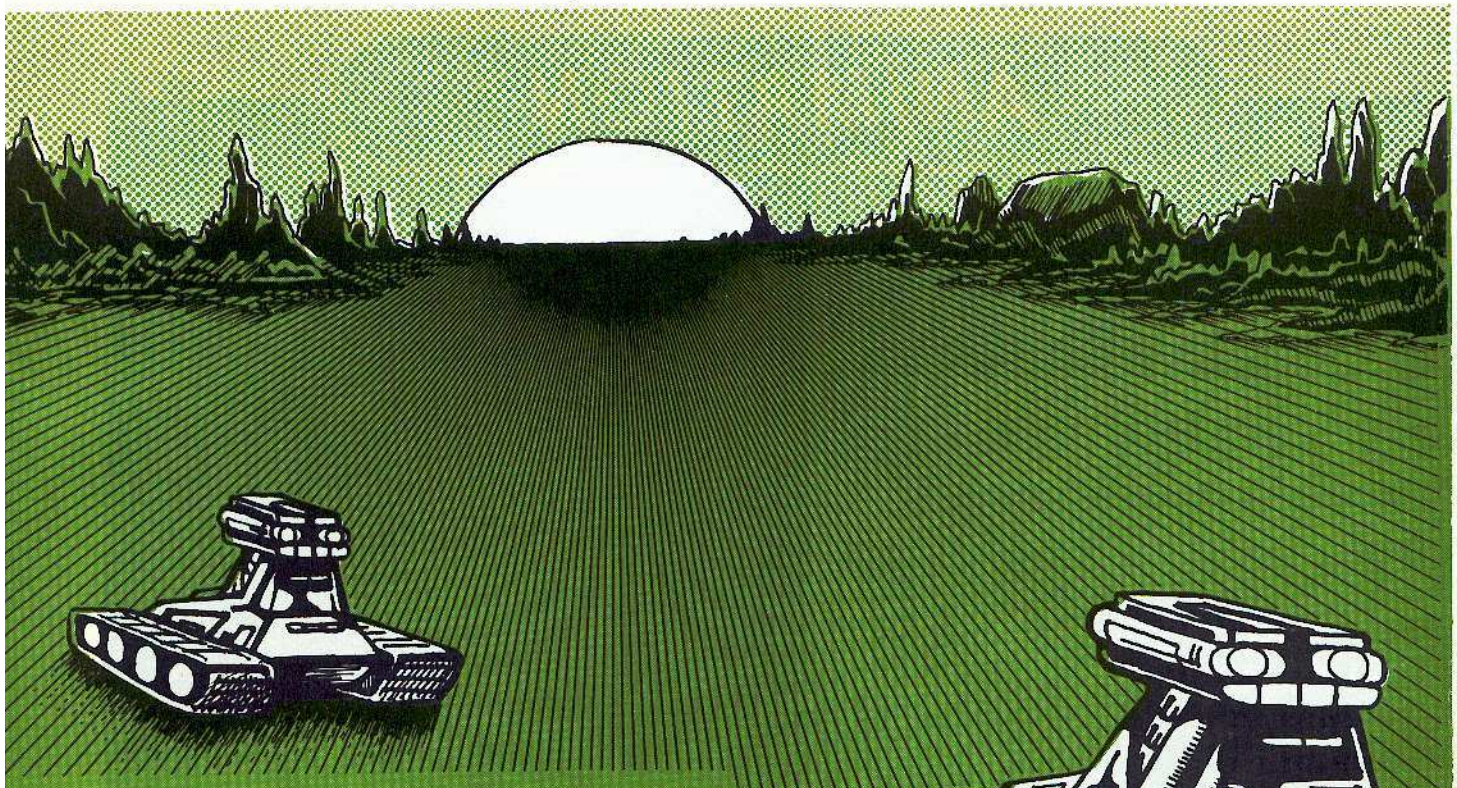
SPACE RAID

REQUIREMENTS:
Quick Draw 1.0
An Apple II+ with Applesoft in ROM.

There are several steps in getting Space Raid up and running:

1. Have *Quick Draw* on disk because the program BLOADs it.

2. Type in the Space Raid Character Set. In this instance, not all 256 character positions are used, only those in the NORMAL set (the ASCII value range of 160 to 223). The easiest way to put them into memory (and then onto disk) is to type in the supplied hex dump. An alternate method (one that I suggest) is to use the *Quick Draw* editor and recreate the shapes from the hi-res illustrations provided. By using the illustrations as a guide, exact replicas can be made, and you can immediately use the editor for something that will



provide a visible result: a game. An additional advantage is that you can make on-the-spot alterations to the character shapes.

3. Save the Character Set (using Q.D. Editor) as ORDE.SPACE RAID.CH

4. Type in the actual game program.

5. SAVE ORDE.SPACE RAID 1.0

Directions For Play

Key	Use
A	travel upward
Z	travel downward
<—	move left
—>	move right
space	shoot!

Game Object:

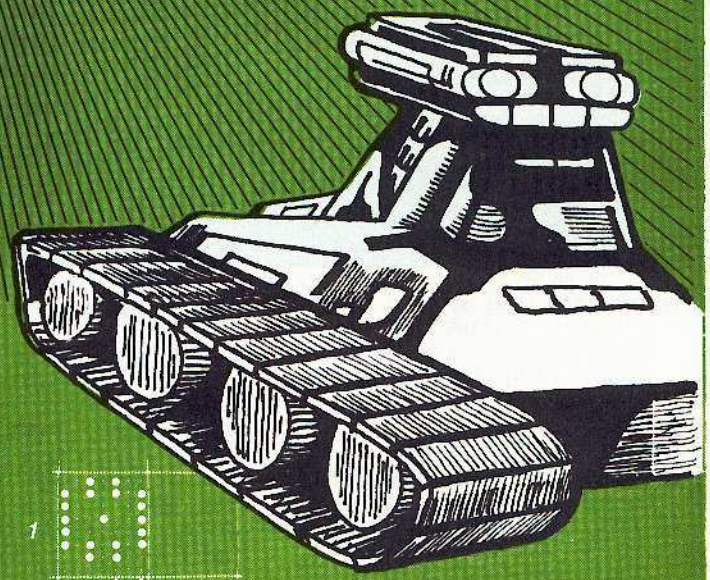
Shoot the raiders (the round shapes) by firing a torpedo from your starship. Your fuel is used whenever you travel. Once you start traveling, your ship will stop only when your torpedo is in motion. Of course, your torpedoes are limited, and when you run out, you have to fly through the apple in order to regain torpedoes. No, it doesn't refuel the ship.

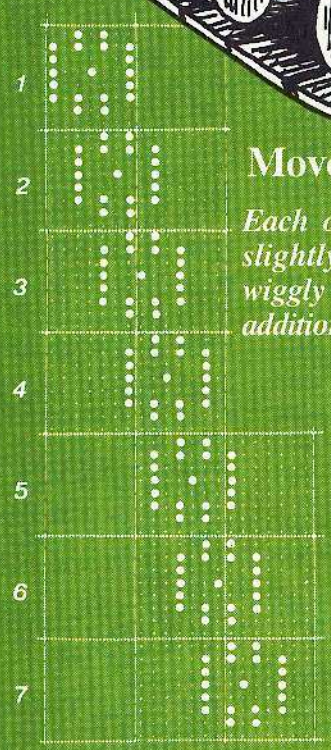
If you shoot all the raiders, you are presented with a still more complex space scene with more raiders and more stars, and your ship is refueled.

Don't shoot the apple. Don't try to fly through a star. And don't run out of fuel . . .

You get three ships. And there are nine skill levels from which to make your selection.

So enjoy.





Move by Seven Shapes

Each of the seven shapes can be slightly different. For example, wiggly legs can be attached for additional animation.

ANIMATION

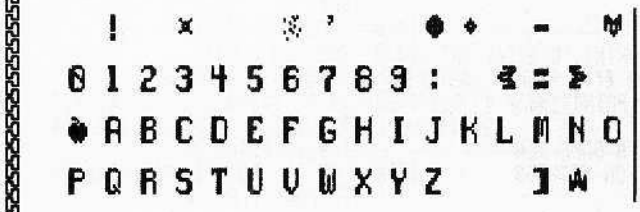

```
10  IF  PEEK (104) * 256 +  PEEK
    (103) <  > 16385 THEN  POKE
    104,64: POKE 103,1: POKE 163
    84,0: PRINT  CHR$ (4)"RUN OR
    DE.SPACE RAID 1.0"
20  REM
LINK QUICK DRAW
30  GOSUB 1360: GOSUB 1520: GOSUB
    1440
40  HOME : RESTORE : GOSUB 1110: GOSUB
    1330: HOME : HGR : POKE  - 1
    6302,0:HT = 0
50 SH = 10:SV = 10:A1 = 0:A2 = 0:
    BO = 200:SP = 3:TP = 10:SC =
    0:S$ = ">": GOTO 90
60  REM
CHECK X,Y
70 L = S(X,Y): RETURN
80  REM
CLEAR MATRIX
90  POKE 109, PEEK (107): POKE 11
    0, PEEK (108): DIM S(40,24)
100  REM
FILL SCREEN
110  FOR I = 1 TO SK * 3 + 8
120  REM
RND X,Y
130 X =  INT ( RND (1) * 38) + 2:
    Y =  INT ( RND (1) * 19) + 4
    : GOSUB 70: IF S(X,Y) OR Y =
    10 THEN 130
140 C$ = "$": IF I > SK * 2 + 5 AND
    I < SK * 3 + 8 THEN C$ = "*"
150  IF I = SK * 3 + 8 THEN C$ =
    "@"
160  VTAB Y: HTAB X: PRINT C$:S(X
    ,Y) =  ASC (C$) + 128: NEXT
170  REM
SET FUEL FOR SKILL
180 FU = 400 + SK * 100:PL = SK +
    2: GOSUB 1090
190  REM
READ KEYBOARD
200 K =  PEEK ( - 16384): IF K =
    160 THEN 680
210  REM
DIRECTION OF SHIP
220  IF K = 136 THEN S$ = "<":A1 =
     - 1:A2 = 0
230  IF K = 149 THEN S$ = ">":A1 =
    1:A2 = 0
240  IF K = 193 THEN S$ = "^":A1 =
    0:A2 =  - 1
250  IF K = 218 THEN S$ = "/":A1 =
    0:A2 = 1
260  VTAB SV: HTAB SH: PRINT " ":
    SH = SH + A1:SV = SV + A2
270  REM
WRAP-AROUND?
280  IF SH < 1 THEN SH = 39
290  IF SH > 39 THEN SH = 1
300  IF SV < 2 THEN SV = 23
310  IF SV > 23 THEN SV = 2
320  REM
SHIP COLLIDED
330 X = SH:Y = SV: GOSUB 70: IF L
     THEN 440
340  REM
NO COLLISION
350  VTAB SV: HTAB SH: PRINT S$: IF
    TP = 0 THEN  POKE 1,10: POKE
    2,5: CALL 768
360  IF A1 = 0 AND A2 = 0 THEN 20
    0
370  REM
DECREMENT FUEL & CHECK
380 FU = FU - 5: VTAB 1: HTAB 30:
     PRINT FU" ": IF FU < 200 THEN
     POKE 1,230: POKE 2,10: CALL
    768
390  IF FU <  > 0 THEN 200
400  VTAB 1: PRINT ".....YOU
    .ARE.OUT.OF.FUEL....."
410  FOR I = 1 TO 100: POKE  - 16
    336, PEEK ( - 16336): NEXT :
     VTAB SV: HTAB SH: PRINT "&"

420  FOR I = 1 TO 100: NEXT : VTAB
    SV: HTAB SH: PRINT " "
430 FU = 400 + SK * 100: GOTO 530

440  REM
DOCKED WITH STARBASE
450  IF L <  > 192 THEN 500
460 SH = SH + A1:SV = SV + A2:X =
    SH:Y = SV: GOSUB 70: IF L THEN
    500
470  REM
MAKE NOISE
480  FOR I = 20 TO 1 STEP  - 1: POKE
    1,I * 2 + 5: POKE 2,7: CALL
    768: NEXT :TP = 10: GOSUB 10
    90
490  GOTO 200
500  REM
SHIP DESTROYED!
510 C$ =  CHR$ (L): VTAB SV: HTAB
    SH: PRINT "&": FOR W = 1 TO
    170: NEXT : VTAB SV: HTAB SH
    : PRINT C$
```

For an explanation of "Space Raid" see the previous page.

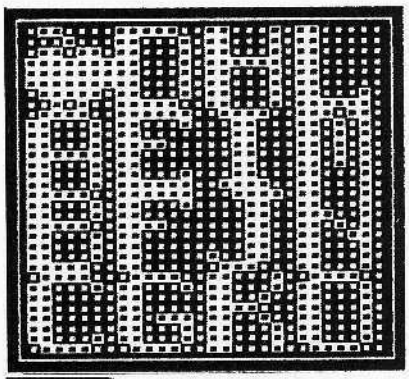
```
520  REM
BOOM!
530  FOR N = 1 TO 100 STEP 8: POKE
     1,N: POKE 2,N: CALL 768: NEXT
     :SP = SP - 1
540  FOR WT = 1 TO 1000: NEXT :S$
      = ">":A1 = 0:A2 = 0:SH = 10
     :SV = 10: POKE  - 16368,0
550  REM
ANOTHER TRY?
560  VTAB 1: FOR I = 1 TO 39: PRINT
     " ";: NEXT : PRINT : GOSUB 1
     000: IF SP THEN 200
570  REM
GAME OVER
580  HGR : POKE  - 16302,0:A$ = "
      GAME OVER ": FOR S = 1 TO 3
     8:P$ = " ": IF S > 14 AND S <
     25 THEN P$ =  MID$ (A$,S - 1
     4,1)
590  VTAB 5: HTAB S: PRINT P$">";
     : IF P$ <  > " " THEN  POKE
     1,S + 3: POKE 2,10: CALL 768
600  FOR J = 1 TO 40: NEXT J,S: HTAB
     39: PRINT " ": POKE  - 16368
     ,0
610  REM
SHOOT LETTERS
620  VTAB 5: HTAB 5: PRINT ">": FOR
     W = 1 TO 1000: NEXT : FOR I =
     16 TO 24: IF I = 20 THEN  NEXT
630  POKE 1,20: POKE 2,40: CALL 7
     68
640  FOR J = 6 TO I - 1: VTAB 5: HTAB
     J: PRINT " +": NEXT J: VTAB
     5: HTAB I: PRINT "&": FOR K =
     1 TO 100: NEXT K
650  VTAB 5: HTAB I: PRINT " ": NEXT
     : VTAB 5: HTAB 5: PRINT " "
660  FOR I = 1 TO 1000: NEXT : GOTO
     40
670  REM
SHOOT TORP
680  POKE  - 16368,0
690  IF A1 = 0 AND A2 = 0 THEN 20
     0
700  REM
ANY TORPS LEFT?
710 TP = TP - 1: IF TP < 0 THEN T
     P = 0: GOTO 200
720  POKE 1,50: POKE 2,60: CALL 7
     68:BH = SH + A1:BV = SV + A2
730  REM
TORP TRAVEL IS 19
740  FOR I = 1 TO 19
750  REM
WRAP-AROUND
760  IF BH < 1 THEN BH = 39
770  IF BH > 39 THEN BH = 1
780  IF BV < 2 THEN BV = 23
790  IF BV > 23 THEN BV = 2
800  REM
HIT SOMETHING?
810 X = BH:Y = BV: GOSUB 70: IF L
      THEN 860
820  REM
PRINT BULLET WITH NOISE
830  VTAB BV: HTAB BH: PRINT "+";
      POKE 1,10: POKE 2,10: CALL
     768
840  REM
ERASE BULLET; MOVE IT
850  VTAB BV: HTAB BH: PRINT " ";
     BH = BH + A1:BV = BV + A2: NEXT
     : GOTO 200

860  REM
HIT WHAT?
870  REM  <HIT STARBASE?>
880  IF L = 192 THEN 1060
890  REM  <HIT SATELLITE?>
900  IF L = 163 THEN  FOR Z = 1 TO
     15: POKE  - 16336, PEEK ( -
     16336): NEXT : GOTO 200
910  REM  <THEN PLANET HIT>
920  REM
NOISE FOR HIT
930  FOR I = 62 TO 1 STEP  - 7: POKE
     1,I: POKE 2,I: CALL 768: NEXT
     : VTAB BV: HTAB BH: PRINT "&
     "
```

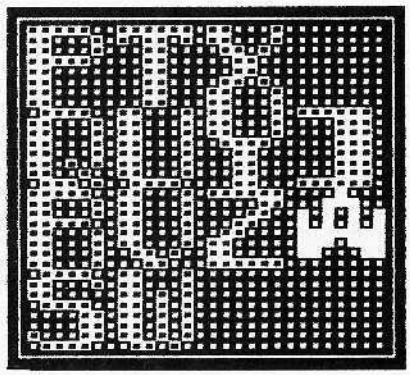
Orde.Space Raid.Ch
(Character Set for Space Raid)

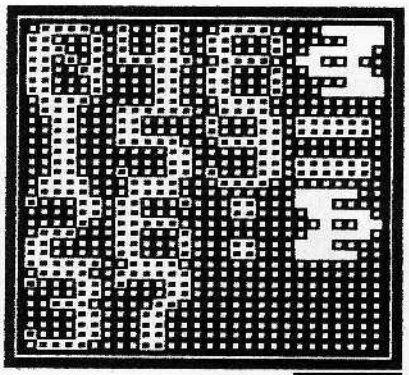
```
1300- 00 00 00 00 00 00 00 00
1308- 0C 0C 0C 0C 0C 00 0C 0C  !
1310- 00 00 00 00 00 00 00 00  "
1318- 00 22 1C 14 1C 22 00 00  #
1320- 00 00 00 00 00 00 00 00  $
1328- 00 00 00 00 00 00 00 00  %
1330- 20 11 04 10 44 09 20 02  &
1338- 18 10 08 00 00 00 00 00  '
1340- 00 00 00 00 00 00 00 00  (
1348- 00 00 00 00 00 00 00 00  )
1350- 1C 3E 6B 7F 6B 3E 1C 00  *
1358- 00 08 1C 36 1C 08 00 00  +
1360- 00 00 00 00 00 00 00 00  ,
1368- 00 00 00 3F 3F 00 00 00  -
1370- 00 00 00 00 00 00 00 00  .
1378- E3 F7 FF D5 D5 9C 88 80  /
1380- 1E 23 27 2B 33 23 23 1E  0
1388- 0E 0C 0C 0C 0C 0C 0C 1E  1
1390- 1E 31 30 10 0E 03 03 3F  2
1398- 1E 31 30 1C 30 30 31 1E  3
13A0- 19 19 19 3F 18 18 18 18  4
13A8- 3F 03 03 1F 30 30 31 1E  5
13B0- 1E 23 03 1F 23 23 23 1E  6
13B8- 3F 31 30 18 0C 0C 0C 0C  7
13C0- 1E 23 23 1E 23 23 23 1E  8
13C8- 1E 31 31 3E 30 30 31 1E  9
13D0- 00 0C 0C 00 00 0C 0C 00  :
13D8- 00 00 00 00 00 00 00 00  ;
13E0- FC F0 BE 93 BE F0 FC 80  <
13E8- 00 3F 3F 00 00 3F 3F 00  =
13F0- 9F 87 BE E4 BE 87 9F 00  >
13F8- 00 00 00 00 00 00 00 00  ?

1400- 06 08 36 7F 7F 7F 3E 14  @
1408- 1E 23 23 23 3F 23 23 23  A
1410- 1F 23 23 1F 23 23 23 1F  B
1418- 1E 23 03 03 03 03 23 1E  C
1420- 1F 23 23 23 23 23 23 1F  D
1428- 3F 23 03 0F 03 03 23 3F  E
1430- 3F 23 03 0F 03 03 03 03  F
1438- 1E 23 03 03 3B 23 23 1E  G
1440- 23 23 23 3F 23 23 23 23  H
1448- 1E 0C 0C 0C 0C 0C 0C 1E  I
1450- 3C 18 18 18 18 18 19 0E  J
1458- 23 23 13 0F 13 23 23 23  K
1460- 03 03 03 03 03 03 23 3F  L
1468- 37 2B 2B 2B 2B 2B 23 23  M
1470- 23 23 27 2B 33 23 23 23  N
1478- 1E 23 23 23 23 23 23 1E  O
1480- 1F 23 23 1F 03 03 03 03  P
1488- 1E 23 23 23 23 2B 13 2E  Q
1490- 1F 23 23 1F 23 23 23 23  R
1498- 1E 23 03 1E 30 30 31 1E  S
14A0- 3F 2D 0C 0C 0C 0C 0C 0C  T
14A8- 23 23 23 23 23 23 23 1E  U
14B0- 23 23 23 23 23 23 16 0C  V
14B8- 23 23 23 2B 2B 2B 2B 16  W
14C0- 23 23 16 0C 0C 16 23 23  X
14C8- 23 23 23 1E 0C 0C 0C 0C  Y
14D0- 3F 30 18 0C 06 03 03 3F  Z
14D8- 00 00 00 00 00 00 00 00  [
14E0- 00 00 00 00 00 00 00 00  \
14E8- 3F 38 38 38 38 38 38 3F  ]
14F0- 08 9C D5 D5 FF F7 E3 80  ^
14F8- 00 00 00 00 00 00 00 00  _
```

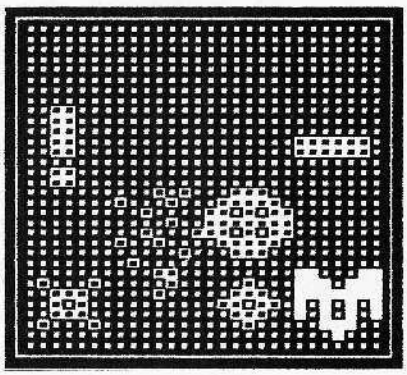


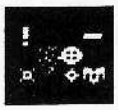
For those of you using the Quick Draw Editor, use this illustration as a guide.











```
940  FOR I = 1 TO 200: NEXT : VTAB
     8V: HTAB 8H: PRINT " "
950  REM
ADD TO SCORE
960 SC = SC + SK * 100:HT = HT +
     1:S(8H,8V) = 0: GOSUB 1090: IF
     HT < PL THEN 200
970  REM
NEW LEVEL
980  REM
ANY SHIPS LOST?
990  IF SP < 3 THEN 1030
1000  HGR : POKE  - 16302,0: VTAB
     10: HTAB 12: PRINT 80" POINT
      BONUS!"
1010 SC = SC + 80: FOR M = 1 TO 3

1020  FOR I = 60 TO 1 STEP  - 5: POKE
     1,I: POKE 2,3: CALL 768: NEXT
     : NEXT :80 = 80 * 2
1030  HGR : POKE  - 16302,0: GOSUB
     1090
1040  REM
STARBASE DESTROYED
1050 SK = SK + 1: HOME :HT = 0:A1
      = 0:A2 = 0:S$ = ">":SH = 10
     :SV = 10:SC = SC + FU: GOTO
     90
1060  FOR I = 5 TO 170 STEP 2: POKE
     1,I: POKE 2,I / 5: CALL 768:
      NEXT
1070  FOR I = 1 TO 1000: NEXT : GOTO
     580

1080  REM
TWILIGHT THEME
1090  VTAB 1: PRINT "SCORE: "SC;:
      HTAB 15: PRINT "SHIPS:"SP;:
      HTAB 24: PRINT "FUEL: "FU"
     ": RETURN
1100  REM
THEME
1110  VTAB 5: HTAB 8: PRINT "@ @
     @  SPACE RAID  @ @ @"
1120  VTAB 7: HTAB 14: PRINT "BY
     RICH ORDE"
1130  VTAB 10: PRINT " A GAME DEM
     ONSTRATING ANIMATION METHODS
     "
1140  VTAB 12: PRINT "USING ROBB
     CANFIELD'S QUICK DRAW UTILIT
     Y"
1150  FOR I = 1 TO 1000: NEXT :I =
     0
1160  REM
WHAT SKILL LEVEL?
1170  VTAB 16: HTAB 8: PRINT "SEL
     ECT A SKILL LEVEL :1-9:"
1180  VTAB 18: HTAB 6: PRINT "1.:
     EASIEST:...TO...9.:HARDEST:"
1190  GOSUB 1210:SK = K - 176
1200  RETURN
1210  DATA 173,48,192,136,208,5,2
     00,1,3,240,9,202,208,245,174
     ,0,3,76,2,3,96
1220  FOR I = 770 TO 790: READ ZZ
     : POKE I,ZZ: NEXT
1230  DATA 77,71,77,97
1240  FOR I = 1 TO 4: READ P(I): NEXT
1250 I = 1
1260 K =  PEEK ( - 16384)
1270  POKE 768,P(I): POKE 769,5: CALL
     770: POKE 768,P(I): POKE 769
     ,26: CALL 770: FOR J = 1 TO
     88: NEXT
1280 I = I + 1: IF I > 4 THEN I =
     1
1290  HTAB  RND (1) * 39 + 1: VTAB
      RND (1) * 5 + 9: PRINT "&"
1300  IF K < 177 OR K > 185 THEN
     1260
1310  RETURN
1320  REM
TWILIGHT THEME
1330  DATA 173,48,192,136,208,6,2
     30,1,198,2,240,8,202,208,244
     ,166,1,76,0,3,96
1340  FOR I = 768 TO 788: READ BY
     TE: POKE I,BY: NEXT
1350  RETURN
1360  REM
OLD COUT
1370 N1 =  PEEK (43604):N2 =  PEEK
     (43605)
1380  REM
QUICK DRAW
1390 N3 = 3:N4 = 8
1400  REM
CHARA
1410 N5 = 0:N6 = 14
1420  RETURN
1430  REM
QUICK DRAW "ON"
1440  POKE 54,N3: POKE 55,N4: CALL
     43114: REM HAVE DOS CONNECT
     QUICK-DRAW
1450  POKE 4,N5: POKE 5,N6
1460  POKE 6,1: POKE 7,1: REM  BL
     OCK SIZE FOR CHARACTER
1470  HOME : HGR : POKE  - 16302,
     0
1480  RETURN
1490  REM
QUICK DRAW "OFF"
1500  POKE 43604,N1: POKE 43605,N
     2
1510  END
1520  REM
LOAD IN FILES
1530  IF  PEEK (2051) = 133 AND  PEEK
     (2052) = 210 THEN  RETURN : REM
     PROGRAM ALREADY IN MEMORY
1540  PRINT  CHR$ (4)"BLOAD QUICK
      DRAW.OBJ"
1550  PRINT  CHR$ (4)"BLOAD ORDE.
     SPACE RAID.CH"
1560  RETURN
1570  REM

SPACE RAID DEMO
1580  REM

WRITTEN BY RICH ORDE
1590  REM
COPYRIGHT 1982
1600  REM
BY SOFTKEY PUBLISHING
```

Screen Cruncher

continued from page 25

Ctrl-D allows you to enter a DOS command such as CATALOG (just type: ctrl-D CATALOG).

Otherwise, the text you enter will be used as the name of a file you wish to load. Once the file is loaded, it will be displayed and compressed. You will see the results of the compression at the bottom of the screen:

1) How many bytes long the compressed picture is.

2) How many bytes were saved.

3) The percentage of savings.

Then you will be asked if you wish to save the compressed version. If you do, press Y for yes and enter its name (again, you may use ctrl-D to enter a DOS command). The suffix .C will be appended automatically to the name of the compressed version of the picture to distinguish it from the original file.

If return is pressed, you will exit this routine without saving the file.

Any other key means "no".

When the option DECOMPRESS (D) is used, you will be asked to enter the name of the compressed file (.C is automatically appended, so do not enter it as part of the file name).

Again, ctrl-D allows you to enter a DOS command.

Pressing RETURN will decompress the currently compressed picture in memory (if there is none, garbage will appear on the screen).

When a picture is decompressed it will appear on the hi-res screen as it is decoded, producing a rather nice scrolling effect from left to right.

To use the UN-PACKer in your own programs, first BLOAD UN-PACK. Then BLOAD the compressed picture, which is normally located on page two of hi-res ($4000). To decompress the picture, enter a CALL 768. By examining the source code you can determine how to load the coded information anywhere, and can control where the decoded picture will be drawn (normally on page one of hi-res, $2000).

DEMO consists of three major sections:

PACK

continued from page 25

```
1830          STX FIRST.TIME.RAN
1840          LDY #$40      RESET STORAGE BUFFER
1850          STY STORAGE+2
1860          LDY #$00      RESET VERT LINE AND
1870          STY GET+1
1880          LDA #$01
1890          STA STORAGE+1 STORAGE BUFFER
1900          LDA #$20      RESET GET BUFFER
1910          STA GET+2
1920
1930
1940 *--------------------------------
1950 * ACTUAL PROGRAM
1960 *--------------------------------
1970
1980 *--------------------------------
1990 *  USE THE FOLLOWING FORMULA TO
2000 * GET THE ADDRESS OF THE LINE
2010 * TO DRAW ON.
2020 *
2030 *--------------------------------
2040
2050 LOOP
2060          TYA
2070          PHA
2080          AND #$C0
2090          STA GET+1
2100          LSR
2110          LSR
2120          ORA GET+1
2130          STA GET+1
2140          PLA
2150          STA GET+2
2160          ASL
2170          ASL
2180          ASL
2190          ROL GET+2
2200          ASL
2210          ROL GET+2
2220          ASL
2230          ROR GET+1
2240          LDA GET+2
2250          AND #$1F
2260          ORA #$20
2270          STA GET+2
2280
2290
2300 *--------------------------------
2310 * STORE THE BYTE ON THE HI-RES
2320 * SCREEN.
2330 *--------------------------------
2340
2350
2360 GET      LDA $FFFF,X  GET A BYTE
2370          BIT FIRST.TIME.RAN FIRST TIME?
2380          BMI .1
2390          STA TABLE    SAVE THIS BYTE
2400          LDA #$80     RESET FIRST.TIME
2410          STA FIRST.TIME.RAN
2420          BMI NEXT
2430 .1       CMP TABLE    SAME AS LAST BYTE
2440          BEQ NEXT     YES
2450          JSR NEW.ONE  SAVE PREVIOUS BYTES
2460 NEXT     INC COUNTER  UPDATE CHARACTER COUNTER
2470          INY
2480          CPY #192     DONE WITH ROW?
2490          BLT LOOP     NO
2500
2510
2520 *--------------------------------
2530 * FINISH OLD BUSINESS
2540 *--------------------------------
2550
2560          JSR NEW.ONE  SAVE CURRENT BYTES IN TABLE
2570          INX          GET NEXT COLUMN
2580          CPX #40      DONE?
2590          BEQ END
2600          LDY #$00
2610          STY FIRST.TIME.RAN RESET COUNTER
2620          BEQ LOOP     ...ALWAYS
2630
2640
2650 *--------------------------------
2660 * NEW.ONE: SAVES THE BYTE AT
2670 * TABLE THE (COUNTER) NUMBER OF
2680 * TIMES. AUTOMATICALLY HANDLES
2690 * REPEATING CHARACTERS.
2700 *--------------------------------
2710
2720
```

I. Relocate itself and load the files.
II. Compress a picture.
III. Decompress a picture.

The first part (I) checks to see if the Applesoft pointers are pointed at $4000 (normally they point at $801). If not, they are modified and the program is re-RUN. Next, the program checks to see if the required files (PACK and UN-PACK) are loaded. If not, they are loaded for you automatically.

The second part (II) will compress a picture by getting the picture's name and calling the Compress Routine ($803, CALL 2651).

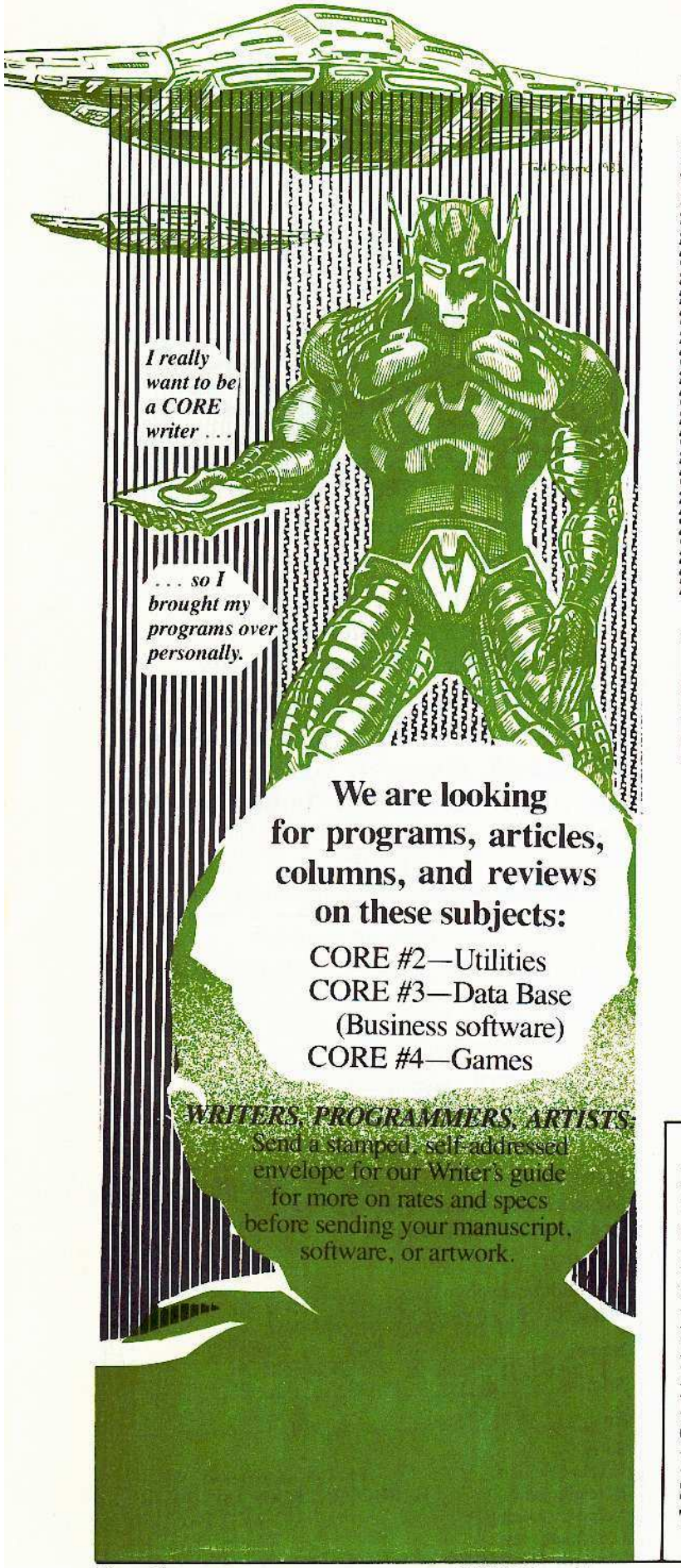
The third section (III) will decompress a picture ($300, CALL 768).

```
2730 NEW.ONE  PHA          SAVE CURRENT.BYTE
2740          STY YSAVE    SAVE Y-REG
2750          LDY COUNTER
2760          CPY #$4      USE REPEAT CHARACTER
2770          BLT NO.REPEAT
2780 REPEAT   LDA REP.CHAR GET REPEAT CHARACTER
2790          JSR STORAGE  AND SAVE IT
2800          TYA          NO. OF TIMES TO REPEATE IN ACCU.
2810          JSR STORAGE  SAVE IT
2820          LDA TABLE    GET CHARACTER TO REPEAT
2830          JSR STORAGE  SAVE IT
2840 EXIT     LDA #$00     RESET COUNTER
2850          STA COUNTER
2860          PLA          RETRIEVE ACCUM.
2870          LDY YSAVE    AND Y-REG
2880          STA TABLE    SAVE CURRENT BYTE IN THE TABLE
2890 END      RTS
2900
2910
2920 NO.REPEAT LDA TABLE   GET BYTE TO REPEAT
2930          CMP REP.CHAR IS IT THE REPEAT CHARACTER
2940          BEQ REPEAT   YES, HANDLE REPEATING CHAR.
2950 .1       JSR STORAGE  AND REPEAT IT Y TIMES
2960          DEY          DONE?
2970          BNE .1       NO
2980          BEQ EXIT     YES SO EXIT
2990
3000
3010
3020 *-------------------------------
3030 * STORAGE: SAVES THE ACCUM. AT
3040 * STORAGE AREA AND INCREMENTS
3050 * THIS VALUE
3060 *-------------------------------
3070
3080 STORAGE
3090          STA $FFFF    ADDRESS TO STORE INFORMATION
3100          INC STORAGE+1 INCREMENT THIS ADDRESS
3110          BNE .1       CARRY?
3120          INC STORAGE+2 YES, SO INCREMENT HIGH BYTE
3130 .1       RTS          RETURN TO CALLER
3140
3150
3160
3170
3180 CURRENT  .BS 1
3190 CURRENT.COUNT .BS 1
3200 LAST.COUNT .BS 1
```

UN-PACK

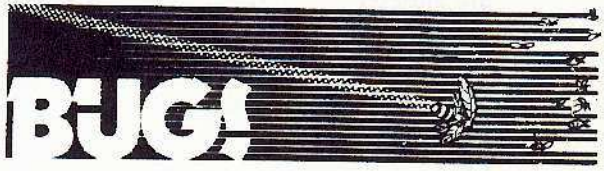
```
1000 *-------------------------------
1010 * HIRES PICTURE UN-PACKER PROGRAM
1020 *
1030 *            BY
1040 *
1050 *        ROBB CANFIELD
1060 *
1070 *       COPYRIGHTED 1983
1080 *     BY SOFTKEY PUBLISHING
1090 *
1100 *-------------------------------
1110
1120
1130
1140 YSAVE    .EQ $1       Y-REG SAVE AREA
1150 COUNTER  .EQ $00      COUNTER FOR REPEATING
1160
1170 REP.CHAR .EQ $FE      MARKER BYTE
1180
1190
1200          .OR $300
1210          .TF UN-PACK
1220
1230
1240
1250          LDX #$00     RESET COLUMN COUNT TO 0
1260          LDY #$40     RESET GET TO $4000
1270          STY GET+2
1280          LDY #$00
1290          STY GET+1
1300          JSR GET      GET REPEAT BYTE
1310          STA REP.CHAR
1320
1330
1340
1350
1360 UN.SCRUNCHER
1370          JSR GET      GET A BYTE TO DECODE
1380          CMP REP.CHAR IS IT THE REPEAT CHARACTER
1390          BEQ DO.REPEAT YES SO RUN THRU REPEAT CYCLE
1400          JSR STORE    NOT REPEAT SO JUST SAVE IT
1410          BCC UN.SCRUNCHER ...ALWAYS
1420
1430 DO.REPEAT
1440          JSR GET      GET NUMBER OF TIMES TO REPEAT
1450          STA COUNTER
1460          JSR GET      GET CHARACTER TO REPEAT
1470 .1       JSR STORE    SAVE IT
1480          DEC COUNTER  KEEP TRACK OF COUNTER
1490          BNE .1       DONE?
1500          BEQ UN.SCRUNCHER YES, SO CONTINUE DECODING
1510
1520
1530 *-------------------------------
1540 * THE GET ROUTINE: GETS A BYTE
1550 * AND INCREMENTS POINTER
1560 *
1570 *-------------------------------
1580
1590
1600 GET      LDA $FFFF    GET A BYTE
1610          INC GET+1    INCREMENT LOW BYTE
1620          BNE .1
1630          INC GET+2    INCREMENT HIGH BYTE
1640 .1       RTS          RETURN TO CALLING PROGRAM
1650
1660
```

```
1670 *-------------------------------
1680 * STORE: STORES ACCUM AT HI-RES
1690 * PAGE AND UPDATES VERT/HORZ
1700 * ADDRESS, ROUTINE FORCES AN
1710 * EXIT TO CALLING ROUTINE IF
1720 * YOU ARE DONE
1730 *-------------------------------
1740
1750
1760 STORE
1770          PHA           SAVE BYTE TO PLACE ON SCREEN
1780          TYA           GET VERTICAL LINE NUMBER
1790          PHA           CONVERT TO AN ACTUAL ADDRESS
1800
1810
1820 *-------------------------------
1830 *  USE THE FOLLOWING FORMULA TO
1840 * GET THE ADDRESS OF THE LINE WE
1850 * WISH TO DRAW ON.
1860 *
1870 *-------------------------------
1880
1890          AND #$C0
1900          STA STORE2+1
1910          LSR
1920          LSR
1930          ORA STORE2+1
1940          STA STORE2+1
1950          PLA
1960          STA STORE2+2
1970          ASL
1980          ASL
1990          ASL
2000          ROL STORE2+2
2010          ASL
2020          ROL STORE2+2
2030          ASL
2040          ROR STORE2+1
2050          LDA STORE2+2
2060          AND #$1F
2070          ORA #$20
2080          STA STORE2+2
2090          PLA
2100
2110
2120 *-------------------------------
2130 * STORE THE BYTE ON THE HI-RES
2140 * SCREEN.
2150 *-------------------------------
2160
2170
2180 STORE2   STA $FFFF,X  SAVE THE BYTE
2190          INY          GET NEXT VERT LINE
2200          CPY #192     DONE WITH COLUMN?
2210          BLT .1       NO SO CONTINUE
2220          LDY #$00     RESET VERT LINE TO 0
2230          INX          GET NEXT COLUMN NUMBER
2240          CPX #40      DONE WITH COLUMNS
2250          BGE .2       YES SO RETURN TO CALLER
2260
2270 .1       RTS          REMAIN WITHIN THIS PROGRAM
2280
2290 .2       PLA          PULL OFF LAST CALLER
2300          PLA
2310          RTS          AND RETURN TO CALLER
2320
```



Get.Obj *continued from page 53*

```
1960 NEXT.LINE STA YP
1970          SEC
1980          LDA XP
1990          SBC #28
2000          STA XP
2010          BNE GET.NEXT.BYTE
2020 *--------------------------------
2030 *
2040 * SET UP FOR SHAPES
2050 *
2060 *--------------------------------
2070
2080 SET.POSN JSR SETHCOL
2090          LDA YP
2100          LDY #00
2110          LDX XP
2120          JMP HPOSN
2130
2140
2150 SAVEALL  STX XSAVE
2160          STY YSAVE
2170          RTS
2180
2190
2200 RESTORE  LDX XSAVE
2210          LDY YSAVE
2220          RTS
2230
2240
2250
```



HARDCORE COMPUTING #3

"Relief Map-Editor"
Page 24, first column.

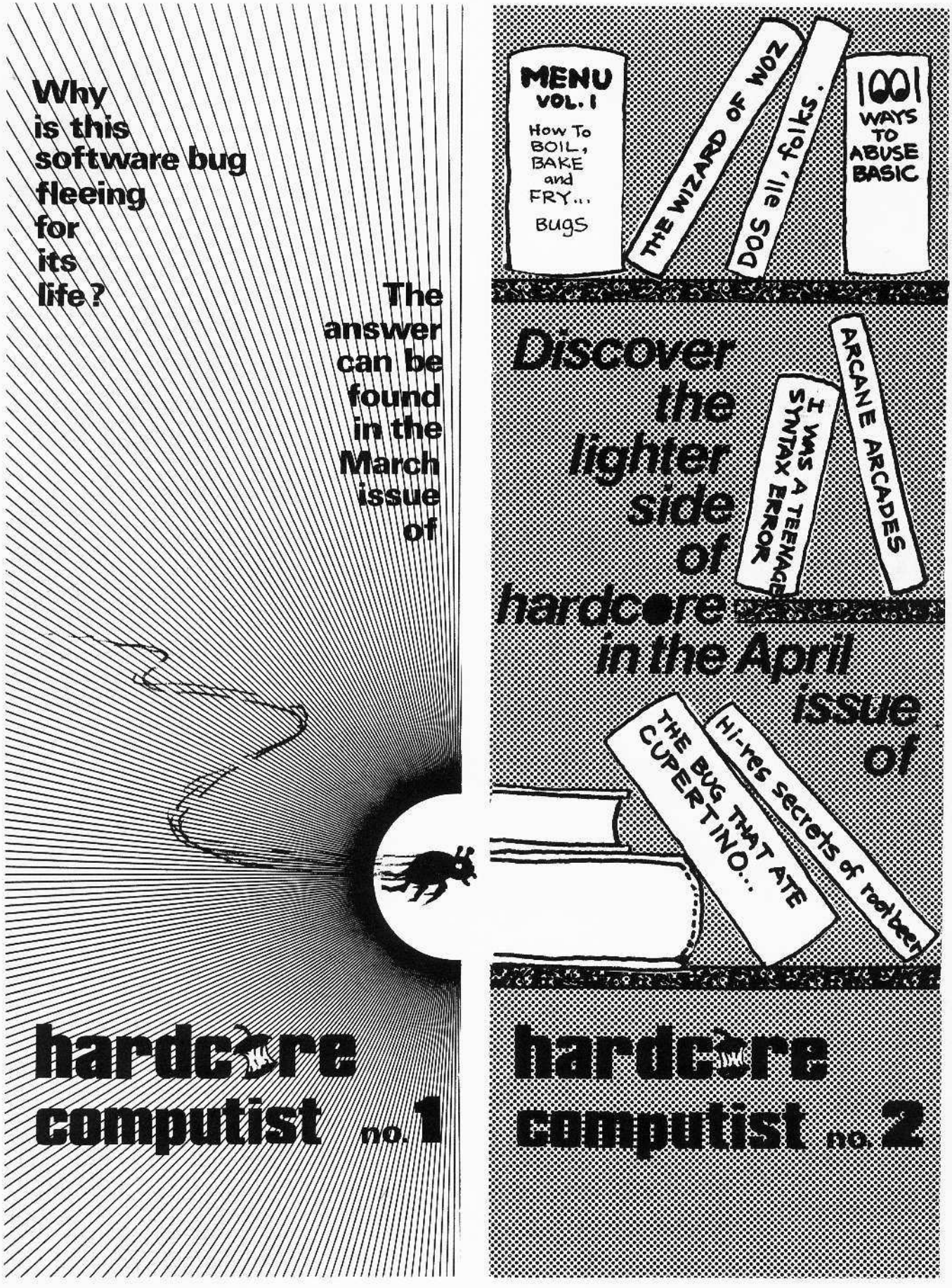
Change:

```
1004Ø AA = BB : BB = BMEM% +
399 : = AA TO BB STEP 2Ø :
GOSUB 11ØØØ : NEXT IN
```

To:

```
1004Ø AA = BB : BB = BMEM% +
399 : TEST = MAX : GAP = -
FOR IN = AA TO BB STEP 2Ø :
GOSUB 11ØØØ : NEXT IN
```

Advertising Index


Why is this software bug fleeing for its life?
The answer can be found in the March issue of
hardcore computist no. 1
MENU VOL. 1
How To BOIL, BAKE and FRY... BUGS
THE WIZARD OF WOZ
DOS all, folks.
1001 WAYS TO ABUSE BASIC
Discover the lighter side of hardcore in the April issue of
I WAS A TEENAGE SYNTAX ERROR
ARCANE ARCADES
THE BUG THAT ATE CUPERTINO...
Hi-res secrets of rodbeer
hardcore computist no. 2

The Sun is dead. All the stars are dead. Voyage to a new universe with

STAR RAMMER

The Earth Ark

Travel through
Newborn Stars,
Throbbing Nebulas,
Exploding Novas,
Spinning Black Holes,
Flashing Pulsars,
Time Tunnels,
Space Warps,
Stargates,
and even Hyper Space.

99 Star Zones

Over 9 BILLION STARGEMS to collect.
WILL YOUR FUEL LAST?
WILL YOUR SHIELDS HOLD UP?
CAN YOU MAKE IT BEYOND ZONE 99?

A FAST-MOVING, FULL-COLOR ACTION GAME WITH ELEMENTS OF STRATEGY AND ADVENTURE

$29.95
Requires 48K and Applesoft ROM

SOFTQUEST
P. O. BOX 44223
TACOMA, WA. 98444