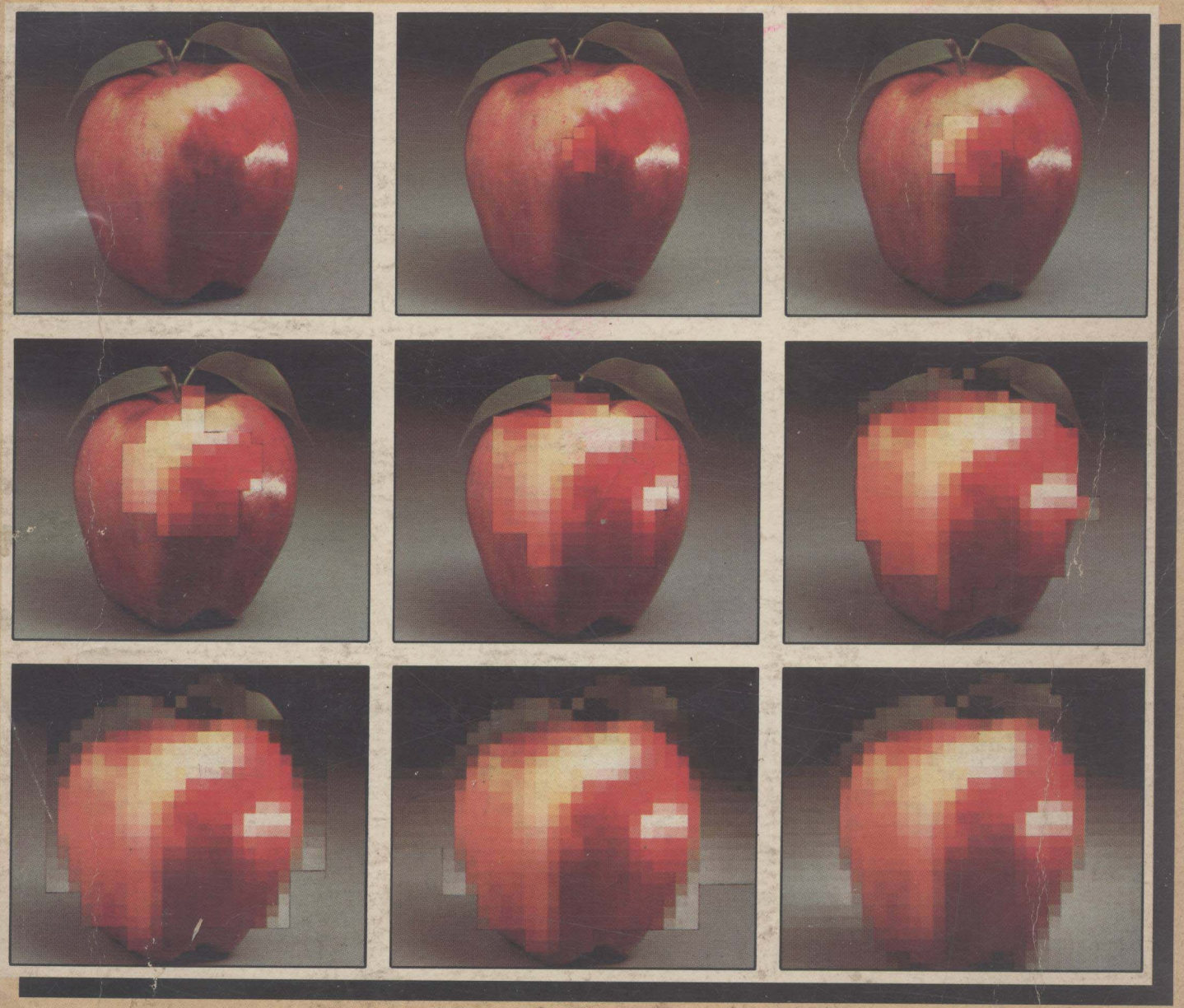
THE CREATIVE APPLE



Edited by Mark Pelczarski and Joe Tate

The Creative Apple

The Creative Apple

Edited by
Mark Pelczarski & Joe Tate

Creative Computing Press
Morris Plains, New Jersey

Library of Congress Number 81-70543
ISBN 0-916688-25-9

Printed in the United States of America

10 9 8 7 6 5 4 3 2

Creative Computing Press
39 East Hanover Ave.
Morris Plains, New Jersey 07950

Introduction

There are dozens of computer magazines on the stands today. But *Creative Computing* was there before Apple (the computer, not its Garden of Eden namesake). *Creative* has followed Apple's development from the original press releases, advertisements, and early versions of its Basic all the way through today's exciting developments with reviews of new products and complete, ready-to-run programs and utilities.

In this book, we've sifted through everything *Creative Computing* has published in the last four years, noting any article we could find that related in some way to the Apple. It wasn't a difficult task. Like thousands of others who read *Creative,* we had wanted to see such a book for a long time. So we set out to put together the book that we always wanted on our shelves.

We love it. We think you will, too. It was a lot of work. We didn't just reprint articles for you. We categorized, updated, verified some of the older sources, and finally settled on well over a hundred articles. These represent the most current and varied cross-section of articles and applications we could find. We used our own Apples to help with the handling of all that information, and for typing and editing the initial manuscript. What a versatile computer!

We've kept all product mentions as current as possible, although any references in the original articles to prices should be used as guidelines, not hard facts. Prices tend to change considerably with competition and demand. We also tried to present a cross-section of what's available, not an absolute, comprehensive list. Although not a "buying guide," the articles about products are a good place to start your search for what you want. Check with your dealers for comparable products and current prices.

This book will give you some idea of the wide range of applications possible with the Apple. Let the possibilities tickle your imagination! There are thousands more that haven't even been touched yet. Don't look at our chapters as absolute boundaries. In categorizing articles we'd often come up with ideas of how something in Chapter X could be used with an idea in Chapter II for an application discussed in Chapter III, and so on.

The most amazing thing about the Apple is the versatility of the machine — how much was designed into it several years back. Although some computers can claim a small degree of superiority in one area or another, none can claim such high marks across a wide range of applications. In other words, if you bought an Apple, it was a choice well made. Here's a book that will help you enjoy your Apple even more and tap its great potential.

Mark Pelczarski & Joe Tate

Contents

Chapter I. Graphics

Chapter II. Music

Chapter III. Education

Chapter IV. Word Processing

Chapter V. Business

Chapter VI. Apple Cart

Contents

Chapter VII. Software Reviews

Chapter VIII. Programs — Ready to Run

Chapter IX. Tips for Easier Programming

Chapter X. Branches

Chapter I
Graphics

Chapter I — Graphics

The first Apples released, back in the days when they only came with Integer BASIC, were touted for their graphic capabilities. But very few people knew how to use anything other than the low resolution (40 by 40) mode. There were special subroutines that you could load from cassette that let you use a high resolution mode, but they were difficult to use and the "hi-res" functions seemed interesting, but limited. Hah!

Those subroutines became part of Applesoft BASIC, and as more and more people started playing with hi-res graphics, more and more tricks and techniques were discovered. State-of-the-art on the Apple now rivals some of the best video arcade games. (Come to think of it, a few years ago "pong" was state-of-the-art at the arcades . . . and the Apple had all that stuff *already built in,* just waiting to be used. Now the six lowly colors provided with Apple hi-res have been mixed and matched into over a *hundred,* and the problem of mixing text and hi-res graphics has been solved many times over. Some computer artists have even claimed that with a digitizing tablet and the software that's available, the Apple is one of the most underrated graphics tools available. It's much more cost-effective than a $200,000 dedicated graphics machine, and in situations where extremely high resolution is not necessary, quite able to take over many of the high-priced systems' functions.

There's been a good cross-section of graphics articles in *Creative Computing,* especially because they run two special graphics issues a year. Included here are some reviews of graphics hardware and software, a few programs and utilities that may enhance your programming or provide some amusement, and articles on some possible interfaces you can use to extend the graphic range of your computer.

Special Notes For Chapter I

- *Apple Sketch,* by David Miller:
  Requires Applesoft, at least 32K of RAM, and a disk drive. It also happens to be attached to a nicely written article.

- *Apple Graphics Utilities,* by David Lubar:
  David Lubar recently reviewed some of the graphics software now available for the Apple. Some other areas of graphics software not covered in this article include the creation of charts and graphs from given sets of data. Dataplot from MUSE and Visiplot from Personal Software are two examples of programs in that category. (Editor's note: see the review of Visiplot at the end of this chapter.) Another would be 3-D animation routines such as 3-D Supergraphics by Paul Lutus (United Software) and Bill Budge's 3-D Game Development Tool (California Pacific). A distinction is made here between 3-D design and animation systems. Design packages can be a lot more flexible because they don't have to adhere to the extremely demanding speed requirements of animation. Of course, the design packages won't let you animate.

- *Picture Packer,* by David Lubar:
  This gives you a neat little pair of machine language programs that will work on any Apple. Even those of you who are not machine language junkies will find the steps behind the development of this utility very interesting as a description of how you can attack a programming problem.

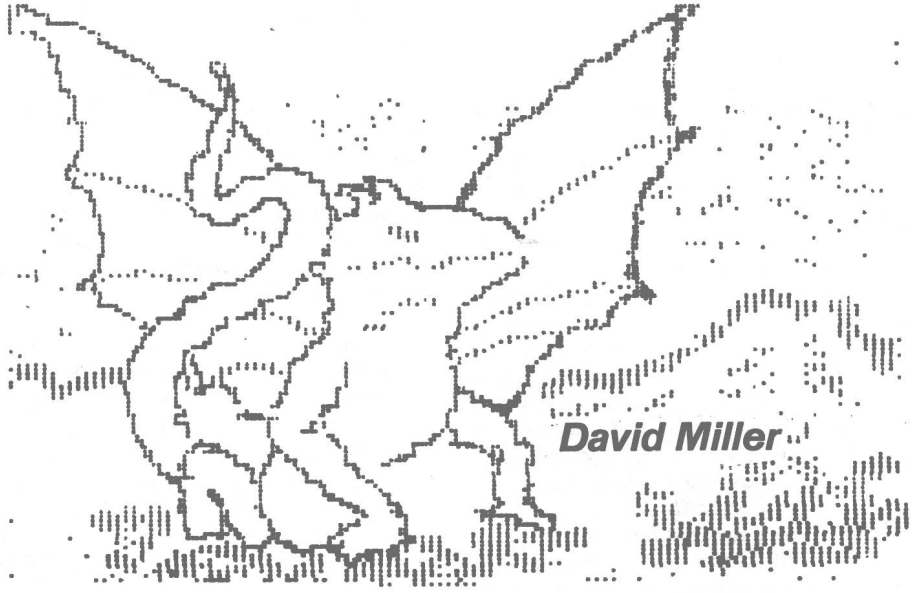
- *Hi-Res Test for the Apple,* by Paul Hitchcock:
  Paul Hitchcock's routines in this article require Applesoft BASIC.

- *Apple II Lo-Res Shape Tables,* by David Lubar:
  This Lo-Res Shape Table routine is written for an Integer BASIC Program.

- *Apple Graphics Tablet; Apple Hi-Res Graphics Made Easy with Versa Writer; Dithering Heights:*
  These three articles describe various ways to put graphic information into your computer, other than with the keyboard, paddles, or a joystick. Of course, the possible input devices are almost endless. Sound, music, almost any kind of motion, or direct electronics from other devices can all be adapted as input. Hmmmm . . .

- Integrating CAI and Video Tape, by Marc Schwartz:
  This last article in Chapter I is about hooking your computer to a videotape machine. This can be done in two ways: the computer controls the videotape machine itself, or the computer video is sent through the videotape machine and taped. Both open a lot of possibilities — from computer aided instruction, as Marc Schwartz suggests — to creating animations on tape. Another area of interest is laser disk technology, which can allow any of thousands of frames to be instantly retrieved, presumably with possible computer interfacing controlling the retrieval. Ah, for a few dollars . . .

Apple-Sketch



When I first bought my Apple II computer a few months ago, I was very excited about using the high resolution graphics that would surely dazzle my friends. I had visions of swooping spacecraft catapulting across the screen, of finely detailed game displays; but most of all, I relished the possibility of drawing freely, in color, on the high resolution screen. What possibilities, I thought: a 45000 point display (give or take a few), with six available colors. Surely there would be an easy way to make it do my bidding! So I spent a few evenings working with my shiny new Apple, a color T.V. set, and the friendly Apple manuals, and quickly learned that things might not be very easy after all.

Apple's high resolution graphics mode consists of two areas or "pages" of memory in which to store the information that makes up a high resolution picture. The primary page picture buffer (or page 1) begins at memory location 8192 and extends up to location 16383, while the secondary page immediately follows, from locations 16384 to 24575. Six colors (black, white, green, violet, orange and blue) are available, although some limitations exist that I will talk more about later. Page one is mixture of graphics and text with four text lines residing beneath the graphics screen, although they can be removed by a simple POKE command. Page two is strictly graphics, with no text window easily available to the user. Basic statements can turn points on or off in color and can draw lines between a specified series of points. For those gamers in the crowd (like myself), binary shape tables can be created to draw, rotate, and expand a user defined shape anywhere on the screen. Sounds great, doesn't it? But let's backtrack for a moment.

To put it frankly, a single point plotted on the screen does not a picture make! A lot of careful thought and planning has to come before you can produce eye-catching drawings using this one-at-a-time method. Since many, many points must be used to produce a colorful drawing of, say, your friendly neighborhood computer, it becomes impractical (and very slow) to use a series of HPLOT statements to draw it from within a program. Who'd want to figure out where all those points should go, anyway? The method used to create shape tables is very tedious and difficult; a single mistake can put you right back where you started from.

As a result, many would-be graphics artists have become frustrated with high-resolution graphics and shy away from using them in their programs. The more affluent, of course, buy expensive graphics tablets costing hundreds of dollars, and produce visual effects to make the rest of us turn green with envy. I soon decided to even up the score a little bit; perhaps through a little software magic, it would become possible to duplicate some of the features of the graphics tablets.

David Miller, 79 Hawley Ave., Port Chester, NY 10573.

The method used to create shape tables is very tedious and difficult; a single mistake can put you right back where you started from.

The result is my program, written in Applesoft to make life a little more enjoyable for the artistic Apple owner. With it you can create vivid computer art, without spending hundreds of dollars on an expensive graphics tablet, by using the Apple paddles for the drawing input. All six colors are available for either the actual drawing or for a change of background color; three pen sizes will produce thick strokes or fine detail. If you'd like to throw in some straight lines between any points on the screen, fine; and it's simple to selectively erase parts of a drawing, or the whole screen, instantly. None of this is going to do you much good if you can't preserve the results of your painstaking labor, so it is possible to save drawings on a disk and load them back into memory another time for viewing or modification. What happens if you get stuck and can't remember how to erase the screen, or change color? Type control-H for help! and a page listing all the commands appears in place of the graphics screen; after perusing for a while, hit a key and the graphics screen is instantly restored, with any drawings intact.

When the program is run, a title page appears, followed by the command page after a key is pressed. As I said, you can pop right back here when in trouble by typing control-H. It's very helpful for beginners, but as you grow more experienced you won't be needing it anymore. Press another key and the real fun starts.

The text screen is now replaced by the black high resolution graphics screen, with a flashing white dot hiding somewhere about. If you doubt my word, turn the paddle knobs and it will magically dance across the screen. This is the point mode, as the first text line informs you, and pen position is indicated by that little blinking dot; it's color is white on a black screen and black on the others to improve visibility. To pop back into it when drawing, press P and the point will reappear. You can use it to carefully erase or chip away at parts of a drawing, or to jump quickly from place to place without drawing.

Ready to start a picture? Hit the W key and the dot stops flashing; by manipulating the paddle, you can produce a fine line of small white dots. Quickly move the pen to separate the dots, or draw slowly and carefully for a smooth white line. You can instantly change colors by pressing the first letter of the color desired (control-B for blue to differentiate it from B for black). It's just as easy to change pen size; press S and numerically choose small, medium, or large by pressing 1, 2 or 3. The point mode resumes, so press a color to begin drawing again in the new size. The pen will now be a small box instead of a single point, and a thicker line will result as you draw. Varying pen size within a drawing

will produce more pleasing results, so use all three to spice up your work. The largest size, by the way, will quickly color in large areas of a picture, such as the sky.

But suppose you'd rather draw black on a white background, or blue on an orange background? No problem! Press X and then the number of the new background color you want; the screen will be instantly filled with that color, and you'll be returned to the point mode. In the process, everything on the screen is erased, so if you hit X by mistake, "return" will get you back to the point mode, safe and sound!

I'm sure that the first few attempts at a drawing will result in a meaningless clutter on the screen; it takes a lot of practice to draw easily with the paddles. By typing control-W, you can erase the whole screen instantly and start over again for (hopefully!) better results. Or you can selectively erase parts of a drawing, either by using the point or with the special erase mode. Press E and a large, blinking square will appear in place of the pen; move it carefully around to erase more quickly than with the point. The eraser is easier to locate than the point, being larger and more visible, so it could also be called a kind of extended point mode.

The line drawing mode can be used to quickly draw straight, even lines between any two points on the screen; a line is drawn in the last pen color used, and its thickness corresponds to the current pen size. After L is pressed, the flashing point reappears on the screen, and its changing X-Y coordinates are continuously displayed in the text window. To set the first endpoint, hit a key; the point will be plotted, followed by a brief pause so that the pen can be moved away. Then the flashing dot reappears, and the second endpoint can be set in the same way. The program asks if there are any corrections, and if the endpoints are exactly where you want them, depress N to draw the line and return to the point mode. Pressing Y gives the option of changing either or both of the endpoints; hit 1 to change point #1, 2 to change point #2, or B to change both. Hitting "return" when in the line drawing mode returns you to the point mode.

Okay, so now you have a great picture done that you want to show your family and friends; how can it be saved? Type control-S and enter the name you'd like it to be stored under; this name should begin with a letter, not a number or control character, although these can be used after the first letter. Hit "return", and the contents of the entire high resolution screen will be saved in a binary file on the disk. Be careful, because an unlocked file with the same name will be erased and written over! If you change your mind about saving, hit "return" to reenter the point mode. The same process holds true for the load command (control-L), but instead of saving, a high resolution picture will be loaded into memory from the specified disk file, gradually filling the graphics screen and erasing anything previously displayed. One important note here: if you load a picture with a colored background, make sure you change the screen to the correct color first. Otherwise, the point mode will start erasing the background, as it might previously have been set for a black screen.

The final command is control-E, and it allows for a graceful, dignified exit from the program, after giving you one last chance to reconsider. More violent types will, I'm sure, use control-C or even (ugh!) reset; make your choice accordingly. And now, for the program itself:

Line 5 resets LOMEM: to 16384, above the primary page of high resolution graphics. While I was entering the program, I painfully discovered that as it grew, it extended upward above location 8192 and into the primary high resolution page, causing the random lines to appear on the graphics screen. The last few lines of the program also acquired bad habits, playing hide and seek with me when more was added to the program. All very unpleasant, but the first line magically cures the problem. Incidentally, LOMEM: is reset to its normal value (just above 2000 with firmware Applesoft), by NEW, DEL, and by adding or changing a program line.

You can instantly change colors by pressing the first letter of the color desired.

Lines 10-50 print the title page, wait for a key to be pressed, clear the keyboard strobe, and go to subroutine 1000, which prints out the command page.

Line 90 traps for an error that might occur during the program run; without it, the program would bomb out, and you would probably lose your picture. It sends program flow to subroutine 4000, where the error is appropriately processed.

Line 100 is initialization; it sets point and screen color, pen size, drawing modes, and D$ as Control-D for DOS commands. It clears the text screen and initializes the high resolution display; then line 105 pops the cursor down to the 21st line directly below the graphics screen and goes to subroutine 650, which handles the point mode and pen size display.

Lines 100-200 handle the drawing, depending on the present program mode and pen size. If a key is pressed, line 110 goes to subroutine 300, which handles commands. Lines 120 and 130 read the two paddles for the X and Y pen position. If the point mode is true, line 140 plots at the present position, erases, resets the color and returns to line 110. Lines 150 and 160 draw a medium and large box in the present color, depending on pen size, if the erase mode is off. If the eraser is on, lines 170 and 180 set its color, depending on the present screen color, and 190 draws and erases it. If none of the above are true, line 200 plots the point and returns to line 110.

The following are program subroutines that are used when commands are input from the keyboard:

Lines 300-385 process commands, calling other subroutines in the process and finally returning to the main drawing section. Line 300 gets the character pressed from the keyboard and clears the keyboard strobe. The remaining lines change colors or modes, or call other subroutines if necessary, depending on the command that has been entered. If the key pressed is not a command, line 385 returns to the main drawing section.

Lines 500-530 save a picture on disk; line 520 alerts DOS with previously initialized D$ and BSAVEs the entire memory contents of high resolution page 1 under the file name input in line 510. A$2000 specifies the starting hexadecimal address (8192 decimal) and L$2000 specifies the length of the area of memory to be stored. Lines 600-630 are used in the same way to load a picture from disk into memory starting at hex location 2000, the beginning of the primary page.

Lines 650-660 set the color of the point for the point mode, depending on the present screen color. Line 660 clears the text window, sets and prints the point mode, goes to subroutine 3000 where the pen size is displayed, and returns.

Lines 700-730 are reached by control-E, and provide a dignified exit from the program. As I said before, killjoys may use "reset" or control-C if they wish.

Lines 800-840 print out the numbers for the different colors and get a character response. "Return" or CHR$ (13) pops you back into the drawing routine after resetting the point mode; an incorrect response causes line 815 to get another character. Otherwise, line 830 sets the new screen color, plots a point, and calls the special machine language routine 62454 to clear the screen in the last color HPLOTed. Finally, line 840 goes to subroutine 650, which handles the point mode restoration and pen size display.

Lines 900-940 use the same method to get a new pen size; a valid response resets

```
PRINT "80       "
]LIST
5  LOMEM: 16384
10  HOME : VTAB 10: HTAB 7: PRINT "***      APPLE-SKETCH       ***": PRINT
20  HTAB 13: PRINT "BY DAVID MILLER"
30  VTAB 19: HTAB 11: PRINT "HIT ANY KEY TO BEGIN"
40  IF  PEEK ( - 16384) <  = 127 THEN 40
50  POKE  - 16368,0: GOSUB 1000
90  ONERR  GOTO 4000
100  HOME : HGR :OLDCLR = 3: HCOLOR= 3:SCRNCLR = 0:PNT = 1:ER = 0:SML = 1:
     MED = 0:LRG = 0:D$ =  CHR$ (4)
105  VTAB 21: GOSUB 650
110  IF  PEEK ( - 16384) > 127 THEN  GOSUB 300
120 X =  PDL (1) * 1.21 + 2: IF X > 277 THEN X = 277
130 Y =  PDL (0) + 2: IF Y > 157 THEN Y = 157
140  IF PNT THEN  HPLOT X,Y: HCOLOR= SCNRCLR: HPLOT X,Y: HCOLOR= OLDCLR: GOTO
     110
150  IF MED AND ER = 0 THEN  HPLOT X,Y: HPLOT X - 1,Y - 1 TO X + 1,Y - 1 TO
     X + 1,Y + 1 TO X - 1,Y + 1 TO X - 1,Y - 1: GOTO 110
160  IF LRG AND ER = 0 THEN  HPLOT X,Y: HPLOT X - 1,Y - 1 TO X + 1,Y - 1 TO
     X + 1,Y + 1 TO X - 1,Y + 1 TO X - 1,Y - 1: HPLOT X - 2,Y - 2 TO X + 2
     ,Y - 2 TO X + 2,Y + 2 TO X - 2,Y + 2 TO X - 2,Y - 2: GOTO 110
170  IF ER AND SCNRCLR = 0 THEN OLDCLR = 3
180  IF ER AND SCRNCLR <  > 0 THEN OLDCLR = 0
190  IF ER THEN  HCOLOR= OLDCLR: HPLOT X,Y: HPLOT X - 1,Y - 1 TO X + 1,Y -
     1 TO X + 1,Y + 1 TO X - 1,Y + 1 TO X - 1,Y - 1: HCOLOR= SCRNCLR: HPLOT
     X,Y: HPLOT X - 1,Y - 1 TO X + 1,Y - 1 TO X + 1,Y + 1 TO X - 1,Y + 1 TO
     X - 1,Y - 1: GOTO 110
200  HPLOT X,Y: GOTO 110
299  REM  *** COMMAND SUBROUTINE
300  GET A$: POKE  - 16368,0
305  IF A$ =  CHR$ (23) THEN  HCOLOR= SCRNCLR: HPLOT 0,0: CALL 62454: HCOLOR=
     OLDCLR: RETURN
310  IF A$ = "W" THEN  HOME : VTAB 21: PRINT "COLOR=WHITE":OLDCLR = 3: HCOLOR=
     3:PNT = 0:ER = 0: GOSUB 3000: RETURN
315  IF A$ =  CHR$ (2) THEN  HOME : VTAB 21: PRINT "COLOR=BLUE":OLDCLR = 6
     : HCOLOR= 6:PNT = 0:ER = 0: GOSUB 3000: RETURN
320  IF A$ = "G" THEN  HOME : VTAB 21: PRINT "COLOR=GREEN":OLDCLR = 1: HCOLOR=
     1:PNT = 0:ER = 0: GOSUB 3000: RETURN
325  IF A$ = "E" THEN  HOME : VTAB 21: PRINT "ERASE MODE":PNT = 0:ER = 1: RETURN

330  IF A$ = "V" THEN  HOME : VTAB 21: PRINT "COLOR=VIOLET":OLDCLR = 2: HCOLOR=
     2:PNT = 0:ER = 0: GOSUB 3000: RETURN
335  IF A$ = "O" THEN  HOME : VTAB 21: PRINT "COLOR=ORANGE":OLDCLR = 5: HCOLOR=
     5:PNT = 0:ER = 0: GOSUB 3000: RETURN
340  IF A$ = "B" THEN  HOME : VTAB 21: PRINT "COLOR=BLACK":OLDCLR = 0: HCOLOR=
     0:PNT = 0:ER = 0: GOSUB 3000: RETURN
345  IF A$ =  CHR$ (12) THEN  HOME : VTAB 21: GOSUB 600: RETURN
350  IF A$ = "S" THEN  HOME : VTAB 21: GOSUB 900: RETURN
355  IF A$ = "P" THEN  HOME : VTAB 21: GOSUB 650: RETURN
360  IF A$ =  CHR$ (8) THEN  GOSUB 1000: GOSUB 650: RETURN
365  IF A$ =  CHR$ (19) THEN  HOME : VTAB 21: GOSUB 500: RETURN
370  IF A$ = "L" THEN  GOSUB 2000: RETURN
375  IF A$ = "X" THEN  GOSUB 800
380  IF A$ =  CHR$ (5) THEN  GOSUB 700
385  RETURN
500  REM   *** SAVE PICTURE
510  PRINT "READY TO SAVE PICTURE    ";: INVERSE : PRINT "RETURN";: NORMAL
     : PRINT " TO EXIT": INPUT "PICTURE NAME ? ";PICTURE$
515  IF PICTURE$ = "" THEN  GOSUB 650: RETURN
520  PRINT D$;"BSAVE ";PICTURE$;", A$2000, L$2000"
530  PRINT PICTURE$;" SAVED.": FOR I = 1 TO 3000: NEXT I: GOSUB 650: RETURN

600  REM   *** LOAD PICTURE
610  PRINT "READY TO LOAD PICTURE    ";: INVERSE : PRINT "RETURN";: NORMAL
     : PRINT " TO EXIT": INPUT "PICTURE NAME ? ";PICTURE$
615  IF PICTURE$ = "" THEN  GOSUB 650: RETURN
620  PRINT D$;"BLOAD ";PICTURE$;", A$2000"
630  PRINT PICTURE$;" LOADED.": FOR I = 1 TO 3000: NEXT I: GOSUB 650: RETURN

649  REM  ***  SET POINT MODE COLOR
650  IF SCNRCLR = 0 THEN OLDCLR = 3: GOTO 660
655 OLDCLR = 0
660  HCOLOR= OLDCLR:PNT = 1: HOME : VTAB 21: PRINT "POINT MODE": GOSUB 300
     0: RETURN
669  REM  ***  CTRL-E  EXIT PROGRAM?
700  HOME : VTAB 21: PRINT "REALLY EXIT THE PROGRAM (Y/N) ? ";
705  GET A$
710  IF A$ = "Y" THEN  TEXT : HOME : GOTO 4100
720  IF A$ = "N" THEN  GOSUB 650: RETURN
730  GOTO 705
799  REM  *** CHANGE BACKGROUND COLOR
800  HOME : VTAB 21: PRINT "BLACK-0   GREEN-1   VIOLET-2   WHITE-3": PRINT
     "ORANGE-5   BLUE-6   ";: INVERSE : PRINT "RETURN";: NORMAL : PRINT "
     TO EXIT"
810  PRINT "NEW BACKGROUND COLOR # ";
815  GET Z$: IF Z$ =  CHR$ (13) THEN 840
820  IF  ASC (Z$) < 48 OR  ASC (Z$) > 54 OR Z$ = "4" THEN 815
830 SCRNCLR =  VAL (Z$): HCOLOR= SCRNCLR: HPLOT 0,0: CALL 62454: HCOLOR= O
     LDCLR
```

pen size and goes to subroutine 650, which handles point mode and pen size display. An incorrect response causes line 905 to get another character.

Lines 1000-1210 display the command page. Line 1000 puts the screen back into the text mode and clears it; after the commands are printed, line 1200 waits for a key to be pressed. Then 1210 clears the keyboard strobe and uses POKE-16304,0 to restore the high resolution graphics screen without clearing it to black (which is what another HGR command would do).

Lines 2000-2470 handle the line drawing, displaying the X and Y coordinates for each point and HPLOTing the lines. Lines 2060 and 2070 read the paddles for point #1, and 2190 and 2200 read them for point #2.

Lines 3000-3030 check for the current pen size and display it beneath the lower right of the graphics screen.

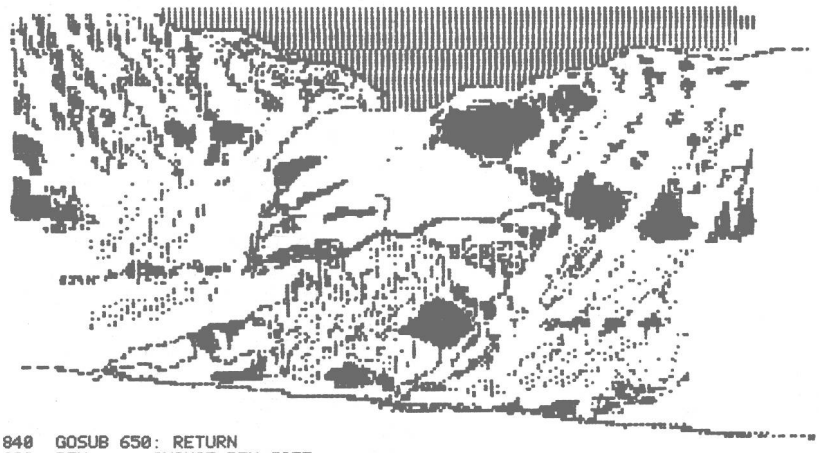
Finally, lines 4000-4070 handle any DOS errors that might occur when loading or saving picutres. Line 4010 sets A to the Apple error code for the trapped

It takes a lot of practice to draw easily with the paddles.

error, which is stored in decimal location 222; it also sets B equal to the line that the error occurred on. Then it clears the text screen and tabs down to the top of the text window, ready to print a message based on the error. The lines after handle:

1. Attempts to load a picture not on disk (DOS "file not found" error).
2. Attempts to save on a write protected disk, such as the DOS master or one with a write protect tab covered.
3. An I/O error, usually encountered when the disk drive door is left open.
4. Attemts to save on a filled disk, or with too little space left to hold the picture.
5. Attempts to save under a file name on disk that is locked.
6. The use of an illegal file name, usually beginning with a number or a control character.
7. Attempts to load a Basic or text file as a picture (believe me, that just won't work!)
8. Control-C: returns the text mode and ends.

As far as I can see, those are the only errors that would normally occur when running the program; all other input is handled by GET and thrown out if inappropriate. If all else fails, line 4060 tells you what error occurred and where before stopping the program. Before



```
840  GOSUB 650: RETURN
899  REM  *** CHANGE PEN SIZE
900  PRINT "READY TO CHANGE PEN SIZE": PRINT "SMALL(1) - MEDIUM(2) - LARGE
     (3) ?";
905  GET A$
910  IF A$ = "1" THEN SML = 1:MED = 0:LRG = 0: GOSUB 650: RETURN
920  IF A$ = "2" THEN SML = 0:MED = 1:LRG = 0: GOSUB 650: RETURN
930  IF A$ = "3" THEN SML = 0:MED = 0:LRG = 1: GOSUB 650: RETURN
940  GOTO 905
999  REM  *** COMMAND PAGE
1000  TEXT : HOME : HTAB 6: PRINT "***   LIST OF COMMANDS   ***": PRINT : PRINT
1010  INVERSE : PRINT "CTRL";: NORMAL : PRINT " B - SETS DRAWING COLOR TO
     BLUE"
1020  PRINT  TAB( 6);"W - SETS DRAWING COLOR TO WHITE"
1030  PRINT  TAB( 6);"B - SETS DRAWING COLOR TO BLACK"
1040  PRINT  TAB( 6);"O - SETS DRAWING COLOR TO ORANGE"
1050  PRINT  TAB( 6);"G - SETS DRAWING COLOR TO GREEN"
1060  PRINT  TAB( 6);"V - SETS DRAWING COLOR TO VIOLET"
1070  PRINT
1080  PRINT  TAB( 6);"P - CHANGES TO POINT MODE"
1090  PRINT  TAB( 6);"E - CHANGES TO ERASE MODE"
1100  PRINT  TAB( 6);"L - CHANGES TO LINE DRAWING MODE"
1110  PRINT  TAB( 6);"X - CHANGES BACKGROUND COLOR"
1120  PRINT  TAB( 6);"S - CHANGES PEN DRAWING SIZE"
1130  PRINT
1140  INVERSE : PRINT "CTRL";: NORMAL : PRINT " W - WIPE SCREEN CLEAR, STA
     RT OVER"
1150  INVERSE : PRINT "CTRL";: NORMAL : PRINT " S - SAVE CURRENT PICTURE O
     N DISK"
1160  INVERSE : PRINT "CTRL";: NORMAL : PRINT " L - LOAD PICTURE FROM DISK
     "
1170  INVERSE : PRINT "CTRL";: NORMAL : PRINT " H - DISPLAY THIS PAGE OF C
     OMMANDS"
1180  INVERSE : PRINT "CTRL";: NORMAL : PRINT " E - EXIT THE PROGRAM"
1190  PRINT : PRINT  TAB( 6);"HIT ANY KEY WHEN READY..."
1200  IF  PEEK ( - 16384) <  = 127 THEN 1200
1210  POKE  - 16368,0: POKE  - 16304,0: RETURN
2000  REM  *** DRAW LINE
2010 LINE$ = "                    "
2020 CH = 0:X1 = 0:X2 = 0:Y1 = 0:Y2 = 0:QX = 0:QY = 0
2030  HOME : VTAB 21: PRINT "LINE DRAWING MODE        ";: INVERSE : PRINT
     "RETURN";: NORMAL : PRINT " TO EXIT"
2040  VTAB 22: PRINT "POINT #1: X=     Y=     <HIT KEY TO SET"
2050  IF  PEEK ( - 16384) > 127 THEN  GET A$: POKE  - 16368,0: VTAB 22: HTAB
     25: PRINT LINE$:QX = X1:QY = Y1: GOSUB 2430: GOTO 2150
2060 X1 =  PDL (1) * 1.21 + 2: IF X1 > 277 THEN X1 = 277
2070 Y1 =  PDL (0) + 2: IF Y1 > 157 THEN Y1 = 157
2075  VTAB 22: HTAB 13: PRINT  INT (X1);
2080  IF X1 < 10 THEN  PRINT "   "
2090  IF X1 >  = 10 AND X1 < 100 THEN  PRINT "  "
2100  IF X1 >  = 100 THEN  PRINT " "
2105  VTAB 22: HTAB 20: PRINT  INT (Y1);
2110  IF Y1 < 10 THEN  PRINT "   "
2120  IF Y1 >  = 10 AND Y1 < 100 THEN  PRINT "  "
2130  IF Y1 >  = 100 THEN  PRINT " "
2140  HCOLOR= OLDCLR: HPLOT X1,Y1: HCOLOR= SCRNCLR: HPLOT X1,Y1: GOTO 2050

2150  IF CH = 1 THEN 2290
2160  IF A$ =  CHR$ (13) THEN 2300
2170  VTAB 23: PRINT "POINT #2: X=     Y=     <HIT KEY TO SET"
2180  IF  PEEK ( - 16384) > 127 THEN  GET A$: POKE  - 16368,0: VTAB 23: HTAB
     25: PRINT LINE$:QX = X2:QY = Y2: GOSUB 2430: GOTO 2280
2190 X2 =  PDL (1) * 1.21 + 2: IF X2 > 277 THEN X2 = 277
2200 Y2 =  PDL (0) + 2: IF Y2 > 157 THEN Y2 = 157
2205  VTAB 23: HTAB 13: PRINT  INT (X2);
2210  IF X2 < 10 THEN  PRINT "   "
2220  IF X2 >  = 10 AND X2 < 100 THEN  PRINT "  "
2230  IF X2 >  = 100 THEN  PRINT " "
2235  VTAB 23: HTAB 20: PRINT  INT (Y2);
2240  IF Y2 < 10 THEN  PRINT "   "
```

running the program, I suggest taking a CATALOG of the disk to see what pictures are available, and which are locked (it would be a good idea to lock an important picture to prevent accidental erasure). Since you won't be able to save onto a locked file name, you'd have to save under another name to preserve your present version without losing the program.

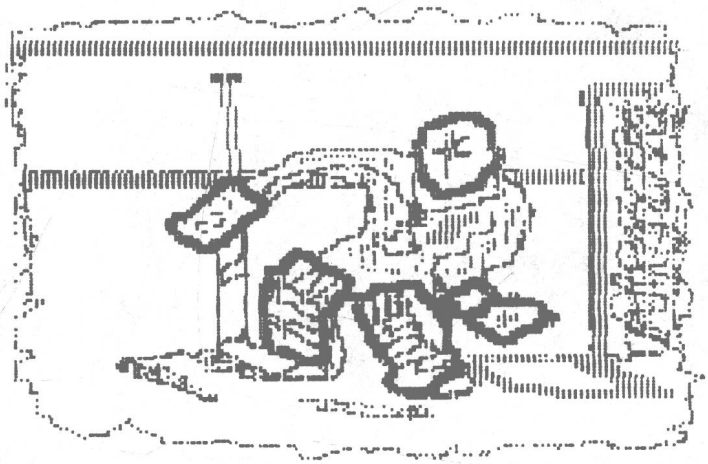
Once a picture has been saved on disk, it can be recalled from a Basic program and displayed using the same technique as in subroutine 600. The program should initialize the string D$ as control-D or CHR$(4), then set the primary page of high resolution graphics and load the picture into memory using DOS commands.

There are a few limitations to what you can do with this program. One problem arises in that the Apple paddles are not very sturdy and tend to become worn with usage. Theoretically, they should produce a steady stream of values from 0 to 255. When new, yes. After a few weeks of Space Invaders and assorted other paddle games, no. Presently, my PDL(O) only goes up to about 180, and PDL(1) to 225. As a result, I use PDL(O) for the Y input, which only has to go to about 160. Unfortunately, the screen is 280 units wide, so my PDL(1) values have to be multiplied to make the pen cover the entire screen, and in the process, some values are lost. With multiplication, I've discovered that about two out of every ten X values just cannot be produced, and it becomes impossible to draw a perfectly smooth line in the X direction. Is there a solution to the problem?

If all else fails, line 4060 tells you what error occurred.

Yes, even if only a partial one, and I suggest it for those who type in this program. The first step is to find out just how your particular paddles will go. Change lines 2060 and 2070 to X1=PDL(1) and Y1=PDL(O). Then enter the line drawing mode and move your paddles clockwise from zero to the highest possible values. Say your paddles are functioning perfectly, and PDL(1) stops at 255. You *can* use the paddle values without multiplying if you leave a small margin on either side of the page. For a perfect paddle, subtract 255 from 280 and divide by 2, leaving a margin of 12.5 on each side. Let's call it 12, and change line 2060 to read:

```
2060 X1=PDL(1)+12:IF X1> 277
     THEN X1=277
```



```
2250  IF Y2 >  = 10 AND Y2 < 100 THEN  PRINT "  "
2260  IF Y2 >  = 100 THEN  PRINT " "
2270  HCOLOR= OLDCLR: HPLOT X2,Y2: HCOLOR= SCRNCLR: HPLOT X2,Y2: GOTO 2180
2280  IF A$ =  CHR$ (13) THEN 2300
2290  VTAB 21: HTAB 1: PRINT "ANY CORRECTIONS ? ";
2295  GET A$
2300  IF A$ =  CHR$ (13) THEN  HCOLOR= SCRNCLR: HPLOT X1,Y1: HPLOT X2,Y2: HCOLOR
= OLDCLR: GOTO 2460
2310  IF A$ = "Y" THEN 2360
2320  IF A$ = "N" AND SML THEN  HPLOT X1,Y1 TO X2,Y2: GOTO 2450
2330  IF A$ = "N" AND MED THEN  FOR I =  - 1 TO 1: HPLOT X1 + I,Y1 + I TO
    X2 + I,Y2 + I: NEXT I: GOTO 2450
2340  IF A$ = "N" AND LRG THEN  FOR I =  - 2 TO 2: HPLOT X1 + I,Y1 + I TO
    X2 + I,Y2 + I: NEXT I: GOTO 2450
2350  GOTO 2295
2360  VTAB 21: HTAB 1: PRINT LINE$;"   ": VTAB 21: HTAB 1: PRINT "POINT 1,
    2, OR BOTH ?";
2370  GET A$
2380  IF A$ =  CHR$ (13) THEN 2300
2390  IF A$ = "1" THEN CH = 1: GOSUB 2470: HPLOT X1,Y1: HCOLOR= OLDCLR: GOTO
    2040
2400  IF A$ = "2" THEN CH = 2: GOSUB 2470: HPLOT X2,Y2: HCOLOR= OLDCLR: GOTO
    2170
2410  IF A$ = "B" THEN CH = 0: GOSUB 2470: HPLOT X1,Y1: HPLOT X2,Y2: HCOLOR=
    OLDCLR: GOTO 2040
2420  GOTO 2370
2430  IF A$ =  CHR$ (13) THEN  RETURN
2440  HCOLOR= OLDCLR: HPLOT QX,QY: FOR I = 1 TO 500: NEXT I: RETURN
2450  HOME : VTAB 21: PRINT "DONE...": FOR I = 1 TO 1000: NEXT I
2460  GOSUB 650: RETURN
2470  VTAB 21: HTAB 1: PRINT LINE$;"     ": VTAB 21: HTAB 1: PRINT "LINE D
    RAWING MODE": HCOLOR= SCRNCLR: RETURN
2999  REM  *** PEN SIZE DISPLAY
3000  VTAB 21: HTAB 25: PRINT "PEN SIZE=";
3010  IF SML THEN  PRINT "SMALL ": RETURN
3020  IF MED THEN  PRINT "MEDIUM": RETURN
3030  IF LRG THEN  PRINT "LARGE ": RETURN
4000  REM  *** ERROR TRAPPING
4010 A =  PEEK (222):B =  PEEK (218) +  PEEK (219) * 256: HOME : VTAB 21
4020  IF A = 6 THEN  PRINT "YOU DON'T HAVE THAT PICTURE ON DISK...": GOTO
    4070
4025  IF A = 4 THEN  PRINT "YOUR DISK IS WRITE PROTECTED !": PRINT "USE AN
    OTHER OR REMOVE WRITE PROTECT TAB.": GOTO 4070
4030  IF A = 8 THEN  PRINT "I/O ERROR...IS YOUR DISK DRIVE OPEN ?": PRINT
    "IF NOT, TRY AGAIN OR USE ANOTHER DISK.": GOTO 4070
4035  IF A = 9 THEN  PRINT "TOO MANY FILES ON DISK; DELETE SOME": PRINT "O
    R USE ANOTHER DISK.": GOTO 4070
4040  IF A = 10 THEN  PRINT "THAT DISK FILE IS LOCKED...UNLOCK IT": PRINT
    "OR USE ANOTHER NAME FOR YOUR PICTURE.": GOTO 4070
4045  IF A = 11 THEN  PRINT "ILLEGAL FILE NAME.": PRINT "PLEASE BEGIN WITH
    A LETTER.": GOTO 4070
4050  IF A = 13 THEN  PRINT "THAT FILE IS NOT A PICTURE...": GOTO 4070
4055  IF A = 255 THEN  TEXT : HOME : GOTO 4100
4060  PRINT "PROGRAM TERMINATED DUE TO ERROR ";A: PRINT "LINE # ";B: GOTO
    4100
4070  FOR I = 1 TO 3000: NEXT I: GOSUB 650: GOTO 110
4100  TEXT : END
```

Now you can get *all* of the X values between 12 and 267, and can draw a perfectly smooth line. You *won't* be able to go outside of those two margins, but they are really very small and I believe the sacrifice is worth it. Likewise, change lines 120 and 2190 to read:

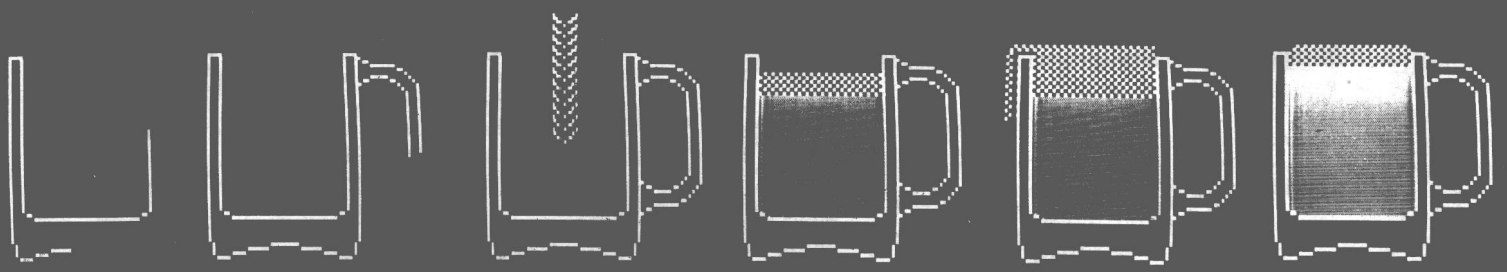
```
120 X=PDL(1)+12:IF X>277 THEN X=277

2190 X2=PDL(1)+12:IF X2>277
     THEN X2=277
```

I've altered my version in the same way; as my paddle goes to 225, I add 27 to the PDL(1) values and lose a somewhat larger margin. It's all up to your personal preference; remember, these are very inexpensive input devices and it's going to take a little experimentation to get them working just right.

I'd also like to warn you that certain colors will not draw cleanly on colored backgrounds, due to the way the high resolution display works. Drawing black on an orange background, for example, produces ragged green fringes, which may also appear when erasing orange on another background. These color fringes appear mainly when using the colored backgrounds. They don't seem to be a problem when drawing on black or white. Consequently, I'd advise drawing on either black or white; if you'd like large areas of another color, switch to the large pen and color them in. Of course, the fringe effects may be just what you want to produce a dazzling, modernistic picture.

The fringe effects may be just what you want to produce a dazzling, modernistic picture.

Don't forget, graphics tablets cost money for a good reason: they are very fast, accurate, and sophisticated. A simple Basic program, such as mine, is hard put to match their performance using very inexpensive paddles as input devices. It can, however, provide a creative challenge and hours of plain old fun if the user has just a little patience and self control (please try not to offend or abuse your poor machine in any way). I can't promise miracles, but believe me, you *can* draw good pictures if you use some imagination and creativity. I know, because I've done some myself that I like very much! Inevitably, some people just won't be able to get the knack of drawing with the paddles, no matter how hard they try. Ah well, they can always go back to playing Space Invaders.... □



Creative Animation

THE CROWD STOPPER

David L. Ross

What do a popcorn popper, diesel engine, birthday cake, beer mug, and a corporate logo have in common? Not much, you say? No so. These items, and countless others, can be computer-animated for display on a color TV screen or monitor in a visually fascinating fashion.

But why would anyone want to animate a popper or beer mug, I hear you cry. It's an area of advertising for which the waters are virtually uncharted. Because people enjoy watching the creative process of image development on the screen, computer animation is an effective promotional tool. Whether the display is placed in a trade show exhibit, store window, building lobby, or permanent product display, the results are impressive. The creative motion on the screen attracts a crowd in a way that a static display or sign seldom does. If cleverly done, animation communicates information to viewers in an entertaining, colorful way. In effect, it's a localized "TV spot" that runs continuously and grabs the attention of passers-by—at a fraction of the cost of regular network or videotape alternatives.

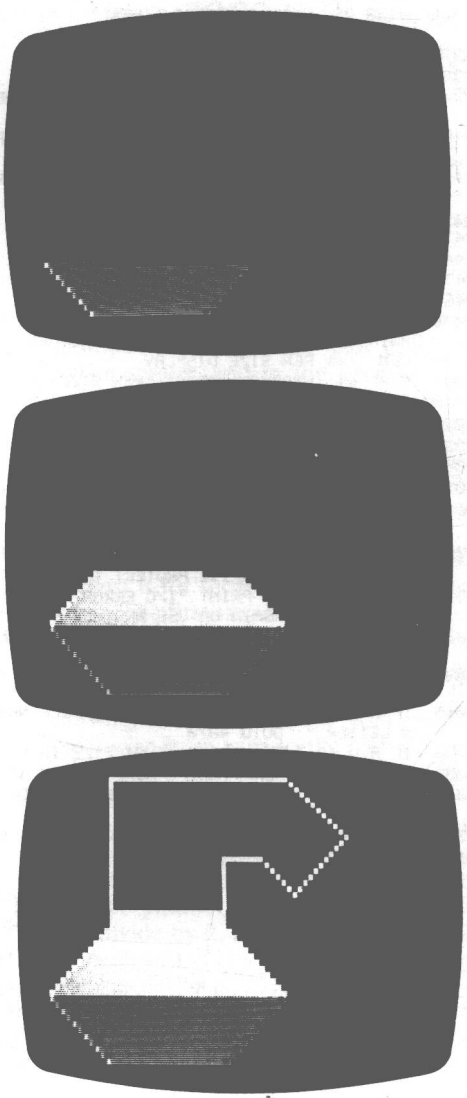
We recognized the universal appeal of the TV screen in early 1979 and began to investigate the possibilities inherent in the use of computer-controlled message displays. At that time, the closest application of this nature was the continuous scrolling of text commonly seen in hotel lobbies and other public areas, announcing meeting rooms, schedules, etc. Such displays are primitive—they're visually boring and can easily be ignored by viewers. Recognizing the potential of and need for more visual impact, we began developing graphic displays to communicate key points, using text only where necessary. Today, our presentations are 70-80% customized color graphics.

We're building a library of animation sub-sequences that can be "dropped into" larger productions—speedboats, champagne glasses, iced cakes with burning candles, flags. We're also developing software vehicles that make program customization, such as the inclusion of corporate logos, a simpler process. A further extension of this concept is the marriage of the video game and advertising presentations, in which the viewer actually participates in the exhibit. This is becoming a powerful "draw," particularly in trade shows. Promotional messages related to the game in play and/or the exhibitor's products and services can be entered from the keyboard, allowing "instant customization," or embedded directly in the program. The originality of this advertising concept, as well as the portability, reliability, and cost of small computers makes the Crowd Stopper an attractive alternative to videotape players or other traditional display devices.

The Popcorn Pumper—Custom Animations

Our popcorn popper animation is summarized in eight frames taken from an animated presentation developed for Wear-Ever Aluminum, Inc., a subsidiary of Alcoa, for trade show display use. Like most animations, it relies heavily on the impact of imaginative development on the screen; it has to be seen in action to fully appreciate the effects. The Popcorn Pumper is drawn on the screen in two colors—yellow and white (just like the real product)—on a black background. The base and chamber outline are built at a variable rate, with

The importance of image development is illustrated in the popcorn popper animation sequence.



David L. Ross, President, Micro Video, P.O. Box 7357, 204E. Washington St. Ann Arbor, MI 48107.