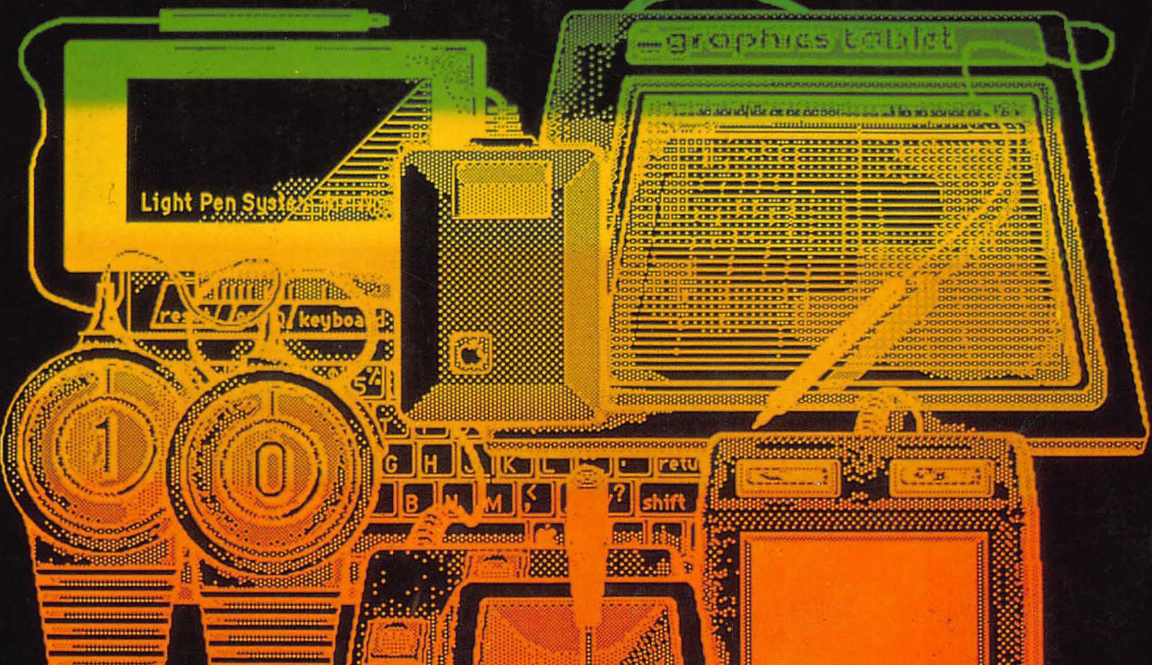
Apple Graphics:

TOOLS AND TECHNIQUES



Michael L. Callery
Roberta Schwartz

Apple Graphics: Tools and Techniques

Michael Callery
Roberta Schwartz

Designed and illustrated by
Roberta Schwartz and Michael Callery

PRENTICE-HALL
Englewood Cliffs, NJ 07632

Library of Congress Cataloging-in-Publication Data

CALLERY, MICHAEL. (date)
Apple graphics.

Includes index.
1. Apple II (Computer)—Programming. 2. Computer graphics. I. Schwartz, Roberta, (date). II. Title.
QA76.8.A66C33 1986 006.6'765 85-19289
ISBN 0-13-039512-9

Editorial/production supervision
and interior design: Barbara Christenberry and Marianne Peters
Cover design: Whitman Studio
Manufacturing buyer: Harry P. Baisley

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-039512-9 01

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Whitehall Books Limited, *Wellington, New Zealand*

Contents

Preface

As teachers of computer graphics, we've been searching for a text to recommend to our students. We wanted a book that would assume no prior knowledge of computers on the part of the reader. We wanted a book that would present information that's accurate, relevant, Apple computer specific (IIplus, //e, *and* //c), and with an emphasis on graphics as they're used by artists. No such book existed. It became clear that to have a text that met our standards we would have to write one.

Apple Graphics: Tools and Techniques is about a family of computers, the Apple II family, that we use with unabashed enthusiasm and success. We designed and wrote the book using a newcomer to the clan, the Macintosh. The layouts, word processing, and illustrations—everything was done on the IIplus, //e, //c or the "Mac". This book is "Apple" from beginning to end.

Each page of this book was written so that when you use it you'll feel as if you have a tutor guiding you. When you're working with a computer and have a question, just find your question and go to the page that answers it. You'll benefit most from this book if you read it from beginning to end, but its uniqueness is in its design as a reference book.

Our emphasis is on graphics, but we begin at the beginning. You must understand how the Apple computer works, and be familiar with Applesoft Basic, if you're to become more than a "doodler".

Programming Apple graphics can be challenging and gratifying, but just as most artists purchase their brushes and paint rather than making their own, most professional computer artists leave the programming to the specialists and use commercial software. We tell you about the packages that are used most frequently and how they are used.

The programs in the book are not space-filling novelties; they supplement the text. Some will serve their purpose after one viewing; others will become valuable utilities for you to use over and over again.

We thank our students; they provided the questions. We hope we've succeeded in providing the answers.

Acknowledgments

We thank:

Edith Namm who entered and tested every program in the book and served as cheerleader during its creation.

Gail Heimberg and her staff at the McGraw Hill Bookstore in New York who graciously allowed us to print the book's illustrations on the Apple laser printer.

Rush Brown, Ame Flynn, Ken Glickfeld, Duke Houston, Lauretta Jones, M. Brooks Jones, Howard Kessler, Lorene Lavora, Maria Manhattan, Jim Markowich, Gideon Nettler, and Catherine Tower who allowed us to reproduce their art and additionally took the time to write biographies and descriptions of their work.

Jay Lininger, Julie Quickel and the rest of the crew at McFarland Graphics and Design, for their cooperation, hospitality, and professionalism.

Bud Therein and Barbara Christenberry of Prentice-Hall for giving us the creative freedom we needed to produce this unique book.

Dedication

In Pennsylvania there lives a lady named Evelyn Callery. In Florida there lives a lady named Clara Schwartz. They don't know each other but they have something in common: us. They are our mothers and we dedicate this book to them.

Why Apple?

At a time when many new computers feature enhanced graphics capabilities, why work with Apple graphics?

Consider that over three million Apple II computers have been sold. More schools and individuals have purchased the Apple to teach and to learn about graphics than any other microcomputer. Although the latest members of the Apple II family have improved features they're largely compatible with the older models.

Consider that for over eight years programmers have been studying the ins and outs of Apple graphics and producing a wide range of graphic tools. Although other computers, including the Commodore 64, Atari 600, 800, and 1200, and IBM PC, have good graphic capabilities—superior to the Apple for some uses—none have the extensive software base of the Apple. Software tools are essential for artists, teachers, and other non-programmers to tap the potential of computer graphics.

Consider that the Apple supports nearly every graphics input and output device in existence.

We've used the Apple for a few years and still there is more to learn and do. We've seen novices of all ages, delighted by its ease of use, draw and paint with this computer. We've seen professional artists, motivated by its unlimited capabilities, create extraordinary images, still and animated.

The Apple computer has proven itself to be an excellent graphics tool.

...then why did you use a Macintosh computer?

We used the Apple Macintosh for word-processing. We could have used an Apple II for this, but no Apple II word-processing program is as easy to use as MacWrite. As we wrote we tested our programs and software on the Apple II—side-by-side computers enabled an interactive writing environment.

We also used the Macintosh to illustrate the book rather than allow our publisher's art department to create illustrations with pencil or pen and ink. We felt it imperative, in a book about computer graphics, to show that all graphics can be generated on a computer.

The Mac allows graphics to be easily integrated into the word processed page. Many of the illustrations could have been done on an Apple II but, as was the case with the text, no Apple program provides the superb combination of high-resolution and ease of use as does MacPaint.

How was this book made?

As we wrote and illustrated the book, we used modems to send each other the text and graphics over standard phone lines.

When the text was complete, it was transmitted from the Mac to a Kaypro computer which was directly linked to typesetting equipment. The text type is Zapf Book Light with Zapf Demi-Bold heads.

The page illustrations were printed with the Apple LaserWriter and the black and white Apple II screen dumps were printed with the Epson printer using Printographer. The color illustrations were reproduced from 35mm slides provided by the artists. Some of these slides were photographed directly off the screen; others were produced from disk files using a Lang slide production system.

Where do I begin?

To create graphics on the Apple computer, begin by learning about your tools.

When you work with more traditional media you don't waste time on the mechanics—if a pencil point breaks you aren't stymied; you sharpen it. When a pen runs out of ink you fill it. You know how to use your tools and you can concentrate on the creative process.

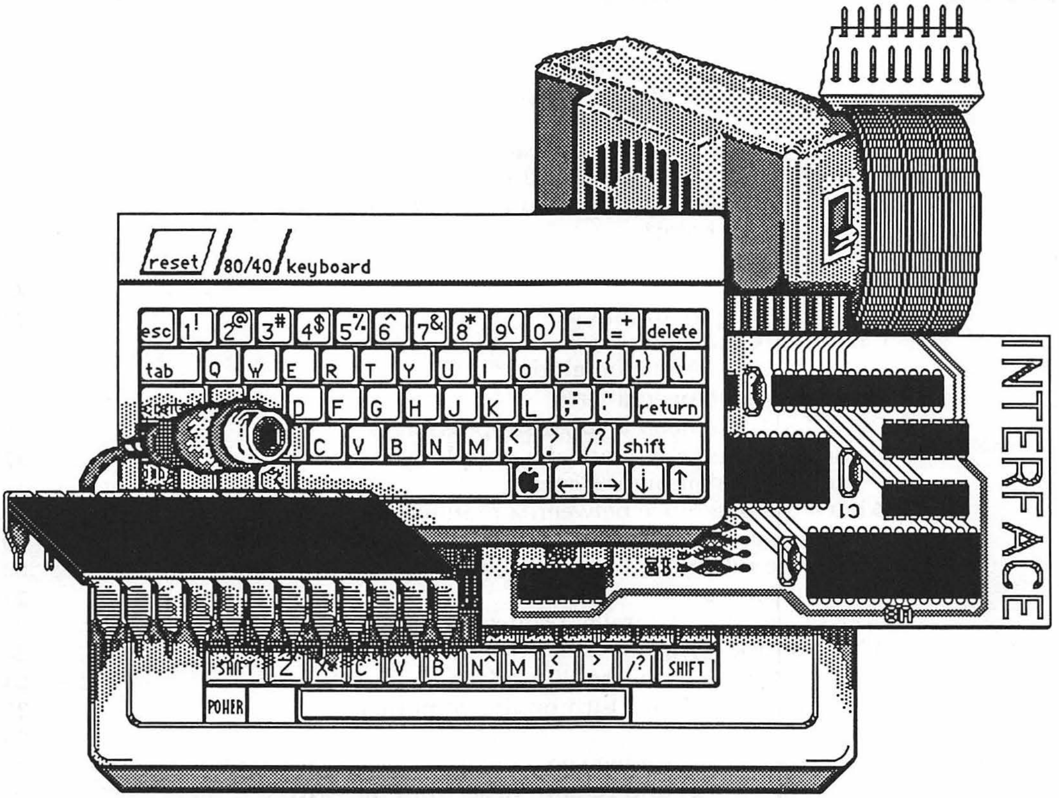
Be just as comfortable with your electronic tools. You're wasting valuable energy if you panic everytime you load a picture and get a "file not found" message. When you're working on a graphic that's due at a client's office the next day, you can't afford to waste time struggling to remember the software's command to invert the image.

Spend time with the computer. You don't have to become a programmer but do key in some simple BASIC programs and see what they do. Initialize disks, load programs, save programs, plot lines on the graphic screens and learn how to save and load images to and from your disks. When you can sit down at the computer and not feel intimidated by it, then, and only then are you ready to concentrate on creating graphics. Working with the computer must become as natural as working with a pencil.

Work with different software packages—get to know each one well before trying another. No graphics utility is difficult to work with if you have a basic understanding of how programs interact with the computer. For the most part all software works the same—the only new thing you'll have to learn with each package is what input (which keys, which buttons, and so on) the program uses.

Remember what each graphics utility does best so that when you work on an image you know which one to use for different effects. Which software has the best color palette? Which program generates the best circles and arcs? Discover the advantages of different input devices. What can you do with the light pen that you can't do with the touch tablet?

You'll find your time was well spent. Master the tools and use them to create masterpieces.



About the Apple Computer

About the Apple Computer

How many Apples are there?

Apple I

Apple Computer Inc. has been producing microcomputers since 1977. Their first computer featured an integrated video circuit with black and white graphics. Until then, a computer's video display was an extra add-on and did not necessarily feature graphics. The first Apple cost $666 and had to be partially assembled by the user.

Apple II

The Apple II computer came entirely assembled with 4K of RAM. Some called it the first "appliance" computer—all it needed was a power outlet and a video monitor. By adding 12K of RAM, Apple II users had four-color (black, green, violet, and white) hi-res graphics. Tinkering with the video display provided another set of colors (black2, orange, blue, white2). By late 1977, Apple was selling only Apple II's with eight-color displays.

The language built into the Apple II was Integer BASIC, written by Steve Wozniak. Integer BASIC was fast but lacked many features needed by programmers: Enter Applesoft. Applesoft, written mostly by Microsoft, is a variant of their standard BASIC. Apple enhanced it, added graphics and game input commands, and sold it on cassette tape. When the Disk II became available in 1978, Applesoft was sold on disk. More and more programs were written in Applesoft and soon Apple provided it in ROM. The Applesoft ROM card, placed in an expansion slot, gave Apple owners instant access to the popular language.

Apple IIplus

By late 1979, Apple had moved the Applesoft ROMs to the main circuit board of the computer (now Integer BASIC had to be loaded from disk or bought on a ROM card). Apple also made modifications to the system monitor to better support a disk drive and dubbed the ROM Applesoft computer the Apple IIplus. For several years, Apple sold the Apple II and the Apple IIplus computer, with a decided sales emphasis on the plus.

Apple ///

In 1980, Apple introduced the Apple ///. This computer was built because the Apple II was thought to be inadequate for many business applications. The Apple /// had a full keyboard, larger memory (128K RAM), and other features, including more and better graphics modes. Many programs written for the Apple II could run on the Apple /// by using an "emulator" program that made the /// behave like a II.

In 1983, Apple introduced the Apple ///plus. Among the improvements in the ///plus were a new video mode with double the vertical resolution from 192 to 384 pixels and 256K of RAM in the standard computer. The Apple /// was discontinued in 1984.

Apple //e

In 1983, Apple upgraded the Apple II to accomodate more memory (128K) and provide a full upper- and lower-case display and keyboard (nearly identical to the original Apple /// keyboard). The Apple //e replaced the II and IIplus. In 1985, the Apple //e was made more compatible with the //c and dubbed the enhanced //e.

Apple //c

In 1984, Apple gave us a transportable version of the //e, the 128K Apple //c. The //c provides many of the features formerly purchased as extras for the II, IIplus, and //e.

Lisa

In January of 1983, a computer unlike all previous Apple computers was introduced. Lisa, geared toward business applications, features a large memory capacity (officially up to 4 Megabytes of RAM) and a new operating environment, the Lisa Office System. Unlike the Apple II and Apple ///, Lisa does not have built-in color graphics, but sports a high resolution, black-and-white display instead. Because Lisa is built around a different microprocessor and has a different operating environment, it is incompatable with software that runs on the Apple II or Apple ///.

In 1985, the Lisa was modified to be more compatible with the Macintosh and renamed the Macintosh XL. The Macintosh XL was shortly discontinued.

Macintosh

In January of 1984, the Macintosh was rolled out. The "Mac" has most of the Lisa features—and a few of it's own—but a different operating system. It has a smaller memory capacity and does not have as many expansion options. Important to graphics people is the fact that the Mac's pixels are square while the Lisa's pixels are rectangular. Screen images transferred between these computers do not maintain their proportions.

Apples Today

The Apple II line consists of the //e and //c. Most Apple II programs run on both of these computers. Most programs will also run on the Apple II and II plus unless they rely on special features of the new generation (like double hi-res graphics).

The Macintosh is currently the sole member of the Apple 32 line. The Mac will not run Lisa software; Lisa can run most Mac software.

How should I set up the computer?

Follow the Apple's guidebooks; they're well written and illustrated.

We can't overemphasize two rules:

1. Turn the computer off before you open the cover or plug into a port—failing to turn off the computer can result in damaged circuit-boards and a dead Apple.
2. Don't unplug the computer when you're plugging and unplugging devices—the system is grounded through the third prong on the computer's plug.

If you need to probe inside the Apple, touch the power supply after opening the case to discharge any static electricity you may have accumulated from carpeting or clothing. (The power supply is the large metal box on the inside left of the Apple's case.) Apple //c users should never need to touch a circuit board.

Don't jam the system into a tight, custom-built cabinet. The electronic components need air to keep cool—a hot system is an unreliable one. (See Does the Apple need a fan? p. 16)

Be careful with the system's cables. Every active computer cable is a tiny radio station and a hodge podge of cables may cause problems. Try to keep cables separate and away from each other. You can use twist-ties to organize them.

If you need extra electrical outlets, use a power strip that features a number of outlets and one or many on-off switches, rather than an extension cord or multiple outlet. Some power strips offer surge supressor circuits and/or line filters. (See Does the Apple need a surge supressor? p. 16) It's best not to hook a computer into the same circuit that runs a heavy-duty electrical appliance like a refrigerator.

Disk drives can be placed on top of the Apple II, II Plus, or //e case or stacked to one side. You may have trouble if you place the monitor too close to the disk drives. Monitors contain magnetic coils, which can damage disks.

Organize the system for comfort and when you work, sit in a comfortable chair that offers proper support for your back.

What are those strange keys?

Computer keyboards resemble typewriters but are actually quite different. Each key, when pressed, produces a code that is stored in a special location in the computer. The keyboard has distinctive key types.

Most keys are just what they seem to be: alphabetic and numeric keys that produce letters and numbers.

Another type of key is a modifier, like SHIFT and CTRL (or CONTROL). By themselves, these keys have no function, but when pressed at the same time as another key, they modify the key value sent to the computer. Pressing A, for example, produces the number 96 on an Apple //e or Apple //c, but if CONTROL is pressed at the same time, the code number 1 is produced. These special control codes are usually used for a specific function. For example, CONTROL H backspaces the cursor and CONTROL G sounds the computer's speaker. Because some of these functions are used frequently, they have keys of their own—the ← and → produce the same codes as CONTROL H and CONTROL U respectively.

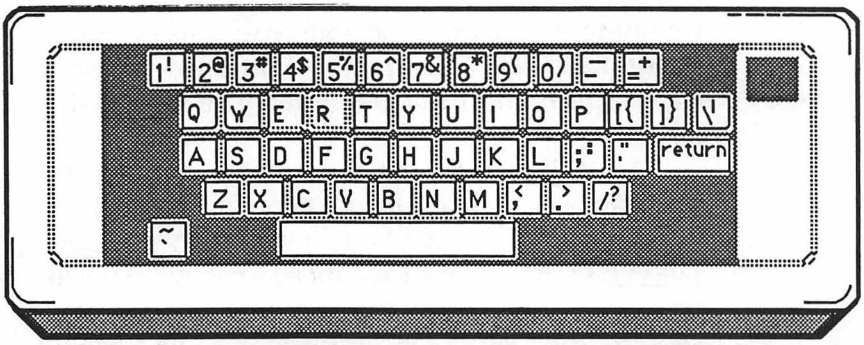
The return key is a confirmation key. As you type, everything is saved in memory, but nothing is interpreted until you press RETURN to tell the Apple that you've finished typing. When you press RETURN any extra characters to the *right* of the cursor are erased. RETURN produces the same code as CONTROL M.

A prefix key is like a modifier—it has no function unless pressed in combination with another key. Unlike a modifier, however, a prefix key is not held down while pressing the second key. On the Apple, prefix keys are used mostly for editing functions. Pressing ESC (or ESCAPE) and then F, for example, clears the screen from the cursor to the end of the screen.

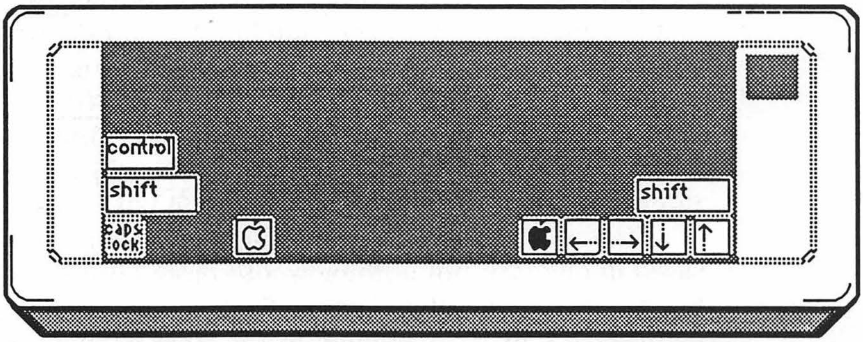
The final type of key on the Apple keyboard is in a category all its own. RESET or *RESET* is a panic button. When you press this key, what happens depends upon which Apple and what program you're using.

Anatomy of a computer keyboard

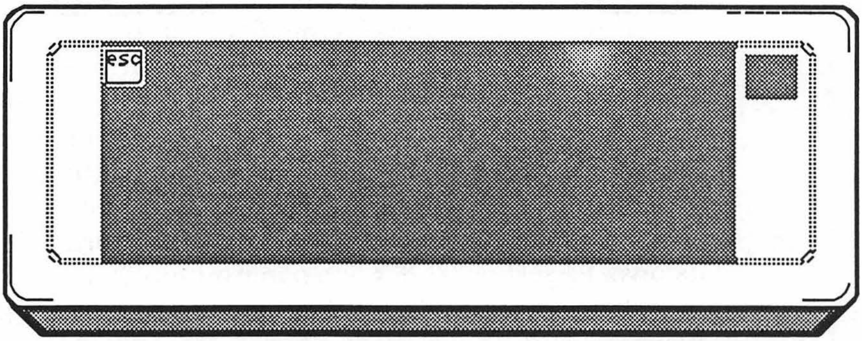
Alphabetic and numeric keys



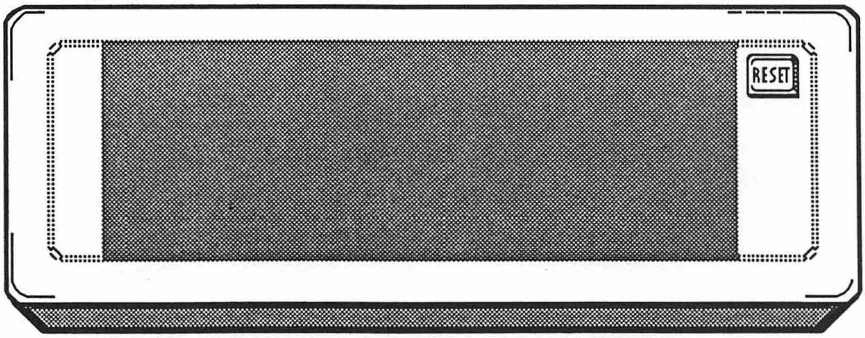
Modifier keys



Prefix key



Reset key



How do I use the Apple II and IIplus keyboard?

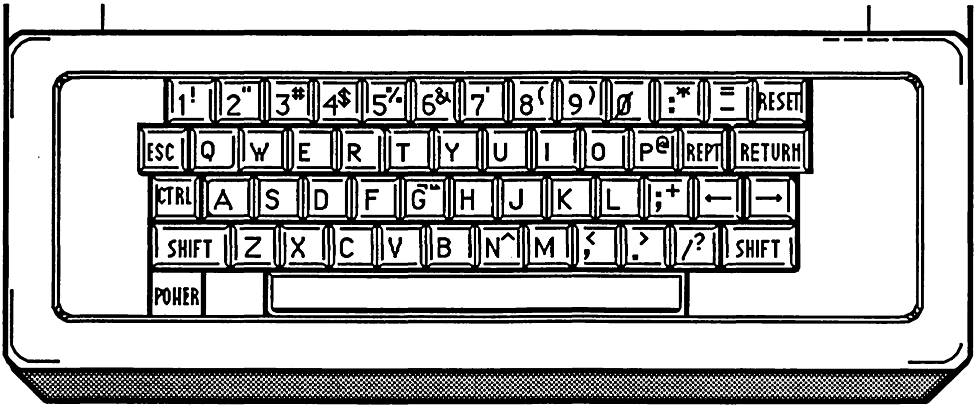
On the Apple II and Apple IIplus, the alphabetic and numeric keys generate upper-case letters or numbers. The shift key functions only where there are two symbols on the keytops. Therefore, typing SHIFT and A produces a capital A but so does an unshifted A. On the other hand, typing SHIFT and P produces the at sign, @, whereas an unshifted P produces a capital P.

There's no easy way to produce lower-case letters on the Apple II and IIplus keyboards although software or hardware modifications can transform the keyboard into a standard upper-and lower-case keyboard. (See Can I get lower-case on the Apple II or IIplus? p. 41)

Be especially careful of RESET. On the Apple II, RESET will stop the program being run and fall into the Apple Monitor. (See What's the system monitor? p. 41) On the Apple IIplus, the RESET key may or may not have an effect depending on the program you're running. Usually, it will stop the program you're running and return you to BASIC or it will reboot the disk.

Apple II and IIplus computers may or may not require pressing the CONTROL key along with the RESET key. The keyboard on most of these computers can be set either way. If the computer you're using doesn't require the CONTROL key, you may be able to change it to do so by throwing a small switch located on a circuit board hanging from the keyboard inside the computer. Remember to turn the computer off before opening the case.

Apple IIplus



1! 2" 3# 4$ 5% 6& 7' 8(9) Ø :* -= RESET
ESC Q W E R T Y U I O P@ REPT RETURN
CTRL A S D F G H J K L ;+ ← →
SHIFT Z X C V B N^ M ,< .> /? SHIFT
POWER

How do I use the Apple //e and //c keyboard?

The Apple //e and //c keyboards are full upper- and lower-case keyboards. You may have to press the CAPS LOCK key when running programs written for the Apple II or Apple IIplus, because many of them won't accept lower-case input. If you're having trouble with a program, check the position of the CAPS LOCK key.

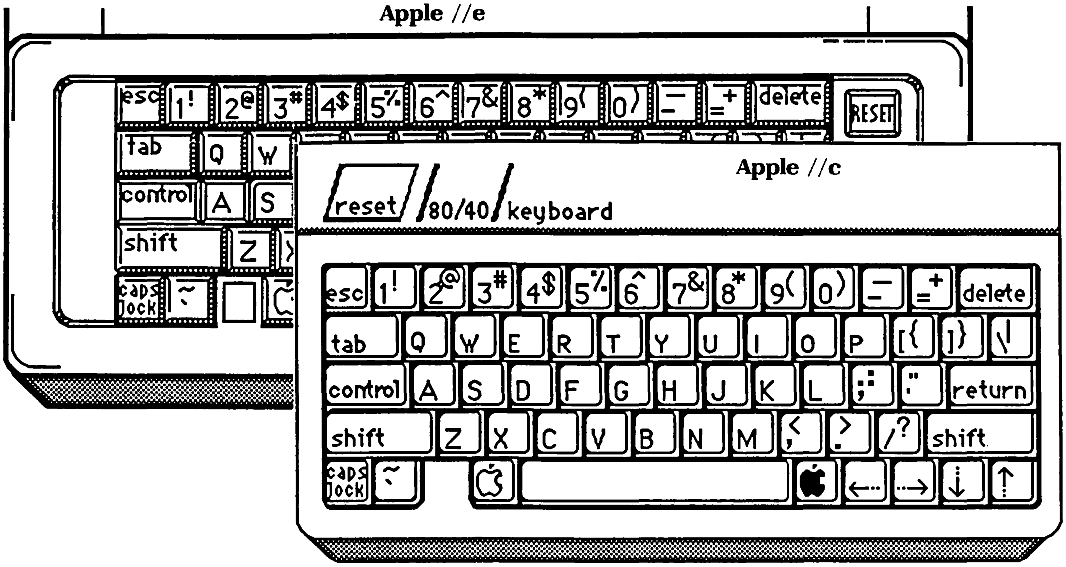
Most special keys, such as TAB, DELETE, ↓, and ↑, won't function unless the program you're running specifically supports them.

Each key generates a unique code, so a program will not accept square brackets where it expects parentheses.

A special three-key sequence can be used on the Apple //e and //c to reboot the disk: CONTROL and 🍏 and RESET.

On the unenhanced Apple //e only, the three-key sequence CONTROL, , and RESET puts the computer into a self-test mode. If everything is OK, you'll see the message "Kernel OK" on the monitor. It'll stay in this mode until you reboot the system.

On the Apple //e and //c, the RESET or *RESET* key must be pressed along with the CONTROL key and may or may not have an effect depending on the program you're running.



What's a peripheral?

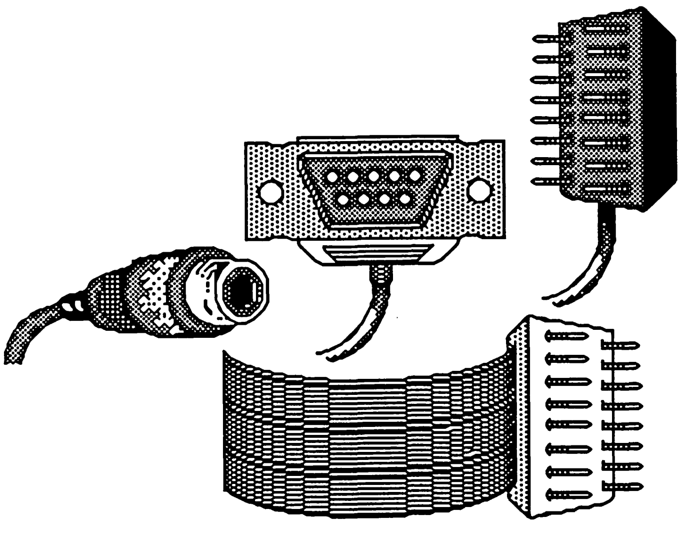
A peripheral is anything that plugs into the computer.

Most peripherals enhance input or output (I/O), the most common being the monitor or CRT, and the disk drive. Printers, joysticks, tablets, and modems are popular peripherals.

Peripherals can be connected to the Apple computer via slots or ports. When connected via the slot, the peripheral uses an interface card. Peripherals that are connected to ports, such as the game port, use pin connectors. The Apple II and IIplus use 16-pin connnectors, while the Apple //c uses 9-pin connectors. The Apple //e has both. Converters are available so that some peripherals with 16-pin connectors can be used with the Apple //c.

The pins at the end of the connectors are fragile—they can bend or break off. When plugging in a peripheral, each pin must go into its designated hole. If a peripheral is not working, check to see if the connector is plugged in upside down. When storing a peripheral, a small piece of styrofoam can be pressed into the pins to protect them.

If you use two or more game port peripherals with the Apple IIplus or //e, avoid opening and closing the lid of the computer by using a port extender. One end of the extender is plugged into the game port and the other end stretches out of the computer (similar to an electrical extension cord). Some extenders let you keep more than one peripheral plugged in at the same time.



What's a slot?

Peripherals must connect to the computer's bus or internal wiring in order to function. On the Apple II, IIplus, and //e, most peripherals are connected via the expansion slots inside the computer. These slots grip the exposed wires through the gold edge connectors on the peripheral's interface card and connect the device to the Apple's bus. (The Apple //c has ports instead of slots. See What's a port? p. 13)

Apple II and IIplus computers have eight slots, numbered 0 through 7. Slot 0 is used to expand the computer's memory from 48K to 64K by the addition of an Apple Language Card or 16K RAM Card. Slot 7 is the only slot that contains the Apple video signal. Therefore, some specialized peripherals, like light pens and RGB Monitor Interface cards, must be plugged into slot 7. Slots 1 through 6 are identical to each other.

Apple //e computers also have eight slots. Seven of these slots are identical to slots 1 through 7 on Apple II and IIplus computers, including the presence of the video signal only on slot 7. There's no slot 0 because Apple //e computers already have a minimum of 64K memory. A special slot called the "Aux (Auxilliary) Slot," can be used for many functions, although it was originally intended for the Apple 80-column card and 64K memory expansion. Interface cards intended for standard slots cannot be plugged into the Aux Slot.

Although the standard slots (except for 7) are identical, tradition has dictated that certain slots be used for particular devices. Install your peripherals in these slots:

Slot	Function
7	RGB video[1]
6	Floppy disk
5	Hard disk
4	Mouse
3	80 column video[2]
2	Modem
1	Printer
0	Memory[3]

[1] RGB Video interfaces are available for Aux Slot on //e, freeing this slot.
[2] Cannot use on Apple //e if there's a card in the Aux Slot.
[3] Not present on the Apple //e. Aux Slot function.

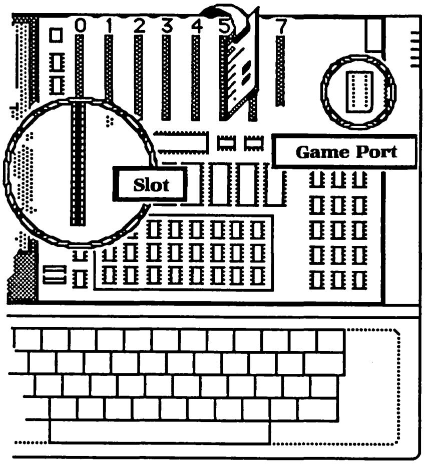
What's a port?

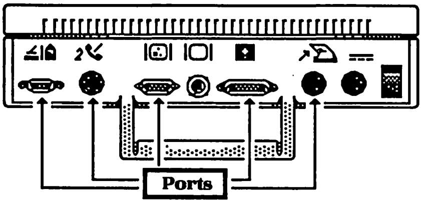
Ports, unlike slots, are merely connectors. The Apple II and IIplus computers have one port. Often called the Game Port, it has several capabilities, including the potential to control external devices. Up to four paddles or two joysticks can be plugged into the game port.

The Apple //e has two game port connectors, but only one game port. One connector, on the inside of the computer case, has a 16-pin plug and contains all game port signals. The other connector, at the rear of the computer, has a 9-pin connector and contains only those signals necessary to drive one joystick or two paddles. You can't plug into both connectors at the same time, since the devices will interfere with one another.

The Apple //c is constructed differently from the II, IIplus, and //e computers. Instead of slots, the //c has many ports. These ports are arranged across the rear of the computer, and when a peripheral is plugged in they behave as if they were slots. To programs being run on the IIc, the built-in 128K acts as if it were an Apple Extended 80-column card in the Aux Slot. The printer, modem, and mouse port behave as slots 1, 2, and 4 respectively. The internal disk and the external disk drive act as if they were plugged into slot 6. The video port accepts RGB video devices. Slots 5 and 7 aren't emulated.

The mouse port on the Apple //c does double duty. When a joystick or pair of paddles is plugged in, instead of a mouse, the port functions like the external game connector on an Apple //e. The extra signals available on the Apple II, IIplus, and //e internal game port are not available on the Apple //c.





How do I plug in an interface card?

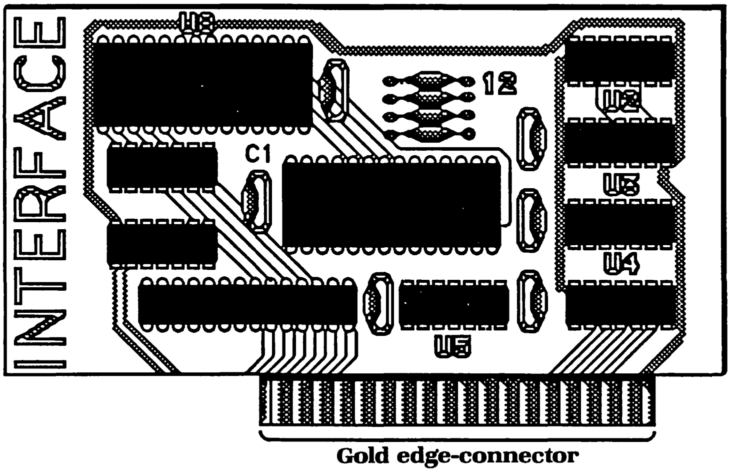
Carefully. Interface cards contain circuits that are sensitive to being scratched or broken and chips that are sensitive to static electricity.

Interface cards usually come in protective shipping material — keep them in this material when they aren't in the computer. If the computer area is prone to static electricity, be sure you're grounded (touch the Apple's power supply to ground yourself) when you touch an interface card.

Hold the card by its edges. Don't touch the interface card's gold-edge connectors. The oils and acids in your skin can reduce the conductivity of the connectors.

Be sure the computer is turned off but still plugged in. (See Can I break the computer? p. 15) Remove the Apple's lid as directed in the owner's manual. Ground yourself. All Apple interface cards—including cards for the Aux Slot on the //e—are designed so that the side containing the chips faces away from the power supply. Gently push the card into the appropriate slot so that the gold-edge connectors make contact with the connectors in the slot. If you must attach cables, be sure to attach them exactly as directed in the instructions that came with the peripheral. Replace the cover and you're ready to go.

With the Apple //c, you've got less to worry about. There are no slots, interface cards, exposed edges, or chips to damage—peripherals are attached simply by plugging them into a port. Turn the power off before plugging into a port.



Can I break the computer?

Yes. You can break the computer if you're careless. Spilling liquids onto the keyboard, dropping it (or the disk drives), plugging it into the wrong current (220V)—these are ways of breaking a computer.

You can damage a computer by plugging or unplugging devices without turning the computer off. Always turn the computer off before inserting a card into an interface slot or plugging a device into a port. Don't unplug the computer—it is grounded via the third prong on the power outlet. Keeping the computer plugged in lessens the chance of damage from static or stray electrical charges.

The power switch is probably the most vulnerable part of the Apple computer. If it breaks, the whole power supply will be replaced. When you take short breaks away from the computer, leave the computer on—don't turn the power on and off. When changing software, use a warm boot rather than turning the computer off and then on again. (See What's booting? p. 26)

Apple computers have excellent service records; they don't break often. Assuming you follow the instructions that came with the system, you are not likely to damage the computer.

Can I leave the Apple on all day?

Some people leave their systems on all the time. There is a slight amount of wear and tear upon the electronic components of the system each time the power is turned on. A normal component should not be harmed by turning the system on and off, however marginal (or flaky) chips may be damaged.

Don't worry about electricity use; personal computers use very little power—comparable to that used by a light bulb.

If you leave the system on for long periods of time between using it, observe these two precautions:

1. Leave a bootable disk in the drive, with the drive door closed. In case of a power outage, this disk will prevent the drive from spinning all night and being damaged. This is not a problem on the //c computers which will display a "Check Disk Drive" message.
2. Turn down the contrast on the monitor to avoid screen burn-in. An unchanging video image can become permanently emblazoned on the screen. The monitor will still work, but the burnt-in image will be distracting.

Does the Apple need a fan?

Fans are used to cool the innards of computers. Apples were designed without fans because convection currents are usually enough to keep the circuits cool and operating correctly. However, if the interior of the Apple is crammed with lots of I/O interfaces or boards—an 80-column card, graphics tablet card, 16K RAM card, and so on—a fan is recommended.

When an Apple overheats, it performs erratically. It may produce nonreproducible errors (errors that are different each time you use the system). This kind of error can be caused by poorly seated chips, corroded interface card connectors, or overheating.

The most popular fans have additional electrical plugs and a master on/off switch built in. By plugging the monitor and printer into these plugs, the one switch will turn the whole system on and off.

Apple //c computers don't need fans. Heat won't cause computer problems if you follow Apple's guidelines—operate the //c only when it's propped up with the handle.

Does the Apple need a surge supressor?

Surge supressors prevent sudden electrical bursts from entering the computer. If you operate the computer in an electrically dirty environment—one where the lights flash or dim, seemingly at random—you may need a surge supressor.

The Apple's power supply is protected against reasonable power fluctuations and it will automatically turn off if the power goes too low or high. This assures the safety of the system, but the data you were working with is lost.

Is the Apple on the same circuit as a refrigerator, electric stove, dishwasher, or other electricity hog? If so, you probably should get a surge supressor. (See How should I set up the computer? p. 6)

What are the components of an Apple system?

An Apple, all by itself, could do very little. Today's microcomputers are not just computers, they are entire computer systems.

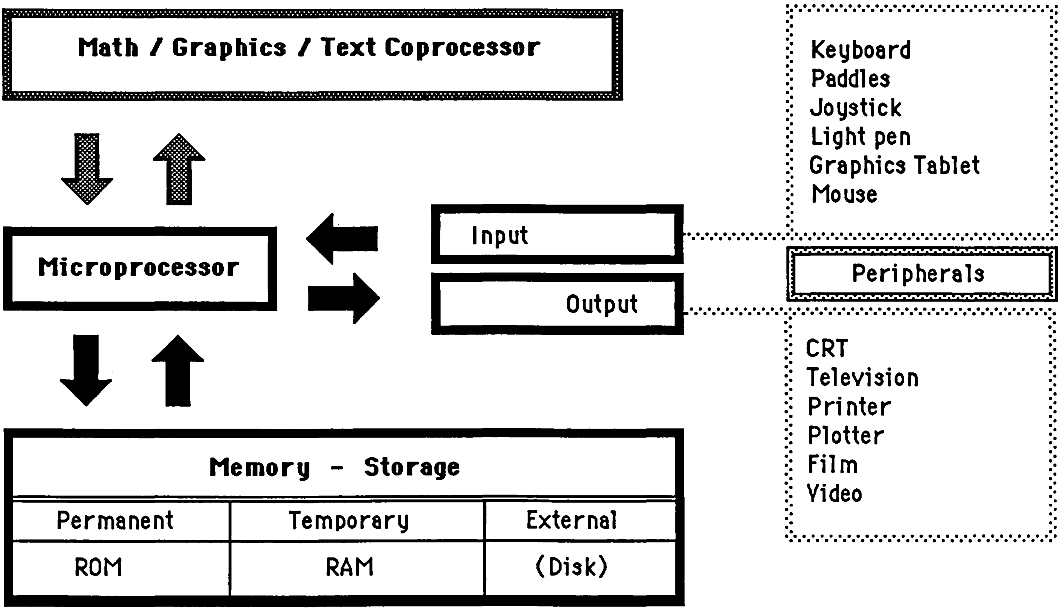
The most important component of the Apple system is the microprocessor or CPU. This device, contained on a tiny silicon chip, is the brain. It is able, among other things, to add, subtract, compare two numbers, and so on. Microprocessors are often named with numbers: the Apple's is 6502 (or 65C02 in the enhanced Apple //e and //c's). The 6502 is termed an 8-bit microprocessor because it processes information in 8-bit chunks. (See What's a byte and a bit? p. 32) The CPU alone can do very little unless it is supported by many other circuits to allow it to store and recall information and to communicate with the outside world.

Information is stored in the computer's memory. We'll have a lot more to say about memory later. (See What's memory? p. 32) The CPU can freely retrieve information from all kinds of memory and can store information in a special type of memory called RAM. Most Apple systems also support external memory, which is not directly available to the CPU, but which can be used for archival or long-term storage. The only practical form of external memory for graphics is disk. With a disk, pictures and programs can be stored, removed from the computer system, and filed for later use. (See What's a disk drive? p. 24 and About disks and files. p. 47.)

Communication is also critical to the operation of a computer system. Information coming into the computer is termed "Input" and information leaving the computer is termed "Output." Computists like to abbreviate this into "I/O." Apples support an astounding array of input and output devices. On the input side, all Apples have a keyboard, and on the output side, all Apples have video output to produce an image on a TV or monitor. Computer graphics artists use perhaps the widest range of input and output devices of any computer users. The Anatomy of a computer system shows some of these. (See What are bytes and bits? p. 35.)

If you reach the limit of the Apple's capabilities or if you are interested in exploring other microprocessors and programming environments, chances are you can find a coprocessor that plugs into an Apple II, IIplus, or IIe. These devices support and extend the inherent capabilities of the 6502.

Anatomy of a computer system



What's the difference between a monitor and a TV set?

For the most part, there's very little difference between the standard, inexpensive monitor (CRT) and a television set. Television sets have tuners which decode electromagnetic broadcast signals back into video signals; monitors don't have tuners.

There is a wide range of monitors, from those that are little more than TVs without tuners to very high-resolution, very costly monitors. The more a monitor diverges from broadcast quality, the more expensive it is.

Inexpensive color monitors use a standard video signal and these are called composite monitors. Higher-resolution monitors invariably require an RGB signal.

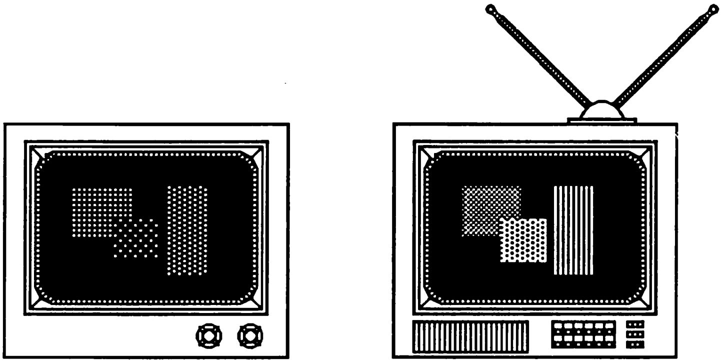
Although there are subtle color differences, Apple computer graphics display equally as well on a color television as on a composite monitor.

Do I need a monochrome monitor if I work in color?

We think you do. It's difficult to read 40-column text (and impossible to read 80-column text) on a composite monitor or a television set, so we work with two monitors—monochrome for text display and programming and color for graphics display.

On a monochrome monitor you can see every pixel on the hires graphic screen; on composite monitors and color television the pixels blend together. The added detail of a monochrome monitor facilitates detailed work such as shape creation and editing graphics.

Furthermore, if you do commercial work, your graphics must look good in monochrome as well as in color because not everyone has a color display. Color combinations that look fine on a color display may be distinguishable on a monochrome display.



What's the difference between composite and RGB?

RGB stands for Red, Green, and Blue. Normally, the term is applied to monitors (CRTs).

RGB monitors require a video signal separated into red, green, and blue components. Standard television signals are termed "composite" because the signal contains all three color signals mixed together in a complex way. In North America, the video signal is called NTSC (National Television Standards Committee). Other parts of the world use PAL or SECAM standards, which are incompatible with NTSC.

Most composite monitors are inexpensive, but they have poor resolution and they can't display legible 80-column text. Graphics look good on these monitors because the colors bleed together and are very saturated.

RGB monitors, on the other hand, don't have to conform to NTSC standard so they can offer higher resolution. RGB monitors can resolve 80-column text. When graphics are displayed on high-quality RGB monitors you can see every pixel in the image.

The Apple II doesn't produce an RGB signal—to generate it you need on RGB interface card that plugs into a slot (usually slot 7) of an Apple II, IIplus, or //e, or the Aux slot of an Apple //e, or the video interface port of the Apple //c. The interface card converts the standard Apple video signal into an RGB signal and enables you to use an RGB monitor.

With the advent of double-high-resolution and its accompanying 80-column text display, RGB monitors are becoming more popular on the Apple. We find that graphics often look better on composite monitors compared to the see-every-pixel display of RGB. Color mixes are especially affected. With a composite display, the colors blend together while an RGB display offers more of a pointillistic image.

What's a printer?

A printer, the most common and necessary external output device for a computer, gives you paper hardcopy.

The computer sends ASCII characters to the printer and the printer reproduces the characters. In this way, you get paper records of disk catalogs, programs, program output, and so on. Printouts are a necessity for debugging, or correcting programs. Virtually all of today's printers are intelligent. That is, they contain a microprocessor, memory, and ROM programs—they're almost computers in their own right.

Many printers are capable of graphics in addition to their normal text mode. To print graphics, a printer must be capable of very precise movement of the print head, daisy wheel, or typing element so that it can randomly print a dot or series of dots on the paper. The printer's capabilities are only half of the graphics equation—there must be a program (a printer dump program) for the Apple to decode the screen.

Printer technology is rapidly changing, but one can broadly characterize printers as impact and non impact. Impact printers make noise; the print head actually hits the paper. Non impact printers are nearly silent. Dot matrix and daisy wheel are the most common impact printers for personal computers.

Impact dot matrix printers print with a matrix of wires, each capable of producing a tiny dot on paper. They're conceptually similar to the video screen, but the aspect ratio of the screen is different from that of the printer. Graphics may be distorted (stretched or squashed) unless a printer dump program compensates for this difference in aspect ratio. The Apple Imagewriter is an impact dot matrix printer.

Color impact dot matrix printers operate identically to their black-and-white counterparts, but with a ribbon banded with colors, and the appropriate software, they produce color printouts. Some colors require overprinting (e.g. red over yellow to print orange), so the ribbon gets muddy with time. Because of this, color ribbons don't last long.

Daisy wheel printers and their offspring, such as thimble wheels, use the period (".") to print graphics. These printers don't produce as fine a graphic as a dot matrix printer, but their typewriter-quality ("letter-quality") text printouts are preferred by many people. To our knowledge, no one produces a daisy wheel printer that can print in full color.

The non impact printers include all of the new as well as some of the most ancient and time-honored printer technologies.

Thermal printers produce low quality printouts; they're dot matrix printers—the printed image is literally burned into the paper by heated wires in the printhead. These printers require special paper that's significantly more expensive than the bond paper used by other printers, and the paper fades with time. Thermal printers, however, are light, compact, quiet, and inexpensive. If expense and transportability are a prime concern, consider a thermal printer.

Thermal transfer printers are similar to regular thermal printers but they use a ribbon between the printhead and the paper. The heat from the dot matrix print head melts the ribbon's ink onto the paper. Thermal transfer printers can use regular paper and produce better, longer-lasting printouts than regular thermal printers. The ribbons don't last long and this technology offers no significant advantage—other than sound level—over regular, impact dot matrix printers. The Apple Scribe is a thermal transfer printer.

Ink jet printers are also in the dot matrix family of printers. Instead of a standard print head or element, an ink jet printer has a tiny nozzle that, upon command, blows a small drop of ink onto the paper. When the ink is used up, the ink reservoir or the entire print head is replaced. Several complicated technologies allow ink jet printers to work, and work they do. Black-and-white ink jet printers are falling rapidly in price and may become as inexpensive as thermal printers. Color ink jets are also reasonably priced, but for full color saturation you need to use special clay-coated paper which absorbs the ink better than bond paper.

Laser printers are the newest and most sophisticated printers. Laser printers are essentially photocopy machines. Instead of scanning a printed page with an electrostatic device onto a drum, laser printers burn the information coming from the computer directly onto the drum. The technology is identical to photocopying. Text printed with a laser printer is nearly indistinguishable from typeset text; graphics can be equally superb with proper software. The Apple LaserWriter has a resolution of 300 dots per inch (dpi) in comparison to 80 or 120 dpi of most dot matrix printers. Laser printers are at the upper end of the cost scale, but they'll probably get cheaper. At this writing, color laser printing is only in the experimental stages, and it'll be far beyond the budget of the average Apple user for some time to come.

How do serial and parallel printers differ?

There's no difference in the actual printing mechanism; the difference lies in the way the printer communicates with the computer.

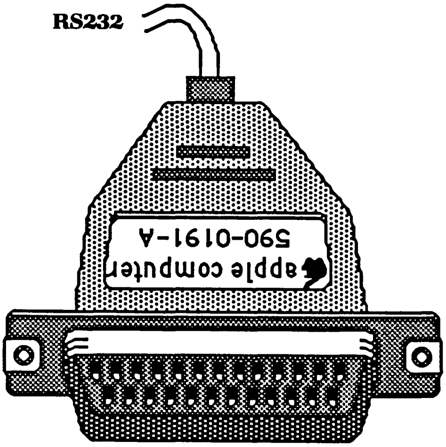
In parallel communication, data is carried in eight wires—one wire for each bit in a byte. A whole byte of data is communicated at once. Although all parallel interfaces work the same way, manufacturers have created several types of parallel interfaces differing in the physical wiring and the use of non-data wires to communicate control information. The most common parallel interface is the Centronics interface.

In serial communication, data is carried in one wire—eight consecutive pulses along the data wire constitute a byte. Although there's only one data wire, many additional wires are used for control information. Serial interfaces are well standardized; the most common serial interface is RS-232. Although building serial peripherals and interfaces is more complicated, the extended control and error-checking capabilities of the RS-232 standard makes it more popular and powerful for certain applications like telecommunications.

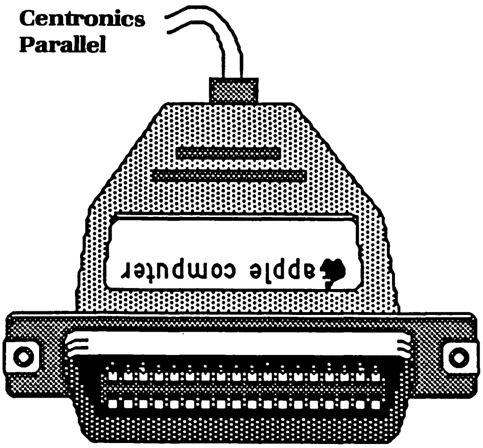
Apple II, IIplus, and //e computers require an interface card that matches the printer's communication circuitry. The Apple //c has two built-in RS-232 serial interfaces. Most Apple peripherals—and all modems—are serial. Parallel peripherals can be connected to the Apple //c with the purchase of a serial-to-parallel translator device.

Most printers are equipped with electronics to accept only one of these communications standards.

RS232



Centronics Parallel



What's a disk drive?

A disk drive contains the mechanism that allows the computer to read from a disk (load data) and write to a disk (save data). The drive contains a motor, spindle, magnetic read/write head, and electronics.

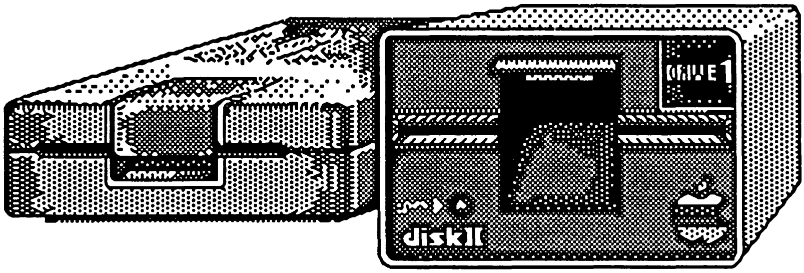
When a disk is inserted into the drive and the door is closed, the disk is firmly held on a rotating spindle, as on a record player. The read/write head hovers over the disk as its underside is supported by a felt pad. When called by the computer, the drive spins and the read/write head moves to the appropriate part of the disk, reading or writing data.

Floppy disk technology is closely related to cassette tape technology with one major difference: disks are random access. To find a particular chunk of information or file on a cassette tape, the computer must start at the beginning and search—linearly—to the end of the tape. On a disk, however, the location of each chunk of data or file is stored as part of the catalog. The computer finds the file by first looking in the catalog and then going directly to the physical location specified in the catalog. Data is not necessarily stored continuously on the disk. The disk operating system can scatter information throughout the disk in small packets; as long as the catalog is intact the file will be retrieved intact.

Do I need more than one disk drive?

Few programs require it, but we recommend a second drive.

With two drives you avoid a great deal of disk swapping: most programs allow you to keep the program disk in drive 1 and your data disk in drive 2. Furthermore, it is extremely important to keep copies or backups of your program and data disks in case a disk is accidentally destroyed. You'll be more likely to make backups if you have a second drive to make it less of a chore.



What happens when I turn on the computer?

When the computer is turned on, all but the earliest Apple IIs run a program that's built into ROM. The program first checks for a disk drive, starting in slot 7 and working down to slot 0. (On the Apple //c, a controller card is always found in slot 6.)

If no disk controller is found, the system initializes BASIC and returns control to you. You can't use disks, but you can enter and run programs. If a disk controller interface card (or the equivalent in a //c) is found, the system runs a program that's stored on this card. This program directs the computer to read the disk that's in the drive.

If there's no disk in the drive (or the disk is unreadable, or isn't a start-up disk), Apple II, IIplus, and //e computers keep the disk drive on, spinning endlessly. Press CONTROL RESET to stop it, or insert a readable disk. (This is one of the few times that it's okay to insert a disk when the drive is spinning.) Apple //c computers display the message "Check Disk Drive," and wait for you to insert a readable disk and press CONTROL RESET.

If there is a readable disk in the drive, the system reads in DOS, ProDOS, or whatever disk operating system is on that disk. This process is termed "booting DOS."

DOS 3.3 now runs the program (usually HELLO) that was specified when the disk was formatted. If this start-up program is deleted from the disk, DOS complains: "File Not Found Error." Ignore the error; DOS is fully functional.

ProDOS loads and runs the system program on the disk. In commercial software, the system program may be the actual application program. On your disks, the system program will most often be BASIC.SYSTEM. This program initializes BASIC and subsequently runs a program called STARTUP. If STARTUP is not present, you are presented with the Applesoft prompt.

Programmers can make an Apple do just about anything when it starts up by rewriting the start-up programs.

If you boot the DOS 3.3 System Master disk, you'll see the BASIC prompt and cursor, indicating that the computer is waiting for you to issue a command. If you boot the ProDOS User's Disk you'll be presented with a menu of choices; you can ask for Applesoft BASIC if you want to program.

What's booting?

Booting is the process of starting up a computer system. Sometimes the term is used in reference to a particular program, for example "Boot DOS" or "Boot up The Graphics Department."

When an Apple system boots, it reads DOS from disk into RAM memory and may execute a start-up program.

The term "boot" is derived from the old saying "Pull yourself up by your bootstraps." The system is learning to read the disk by reading the disk! There's no sleight of hand here, only the edge given by the ROM program on the disk controller card. This program will find and read a small part of the disk—the "boot sectors." The boot sectors contain another program that reads more program segments from the disk until the entire disk operating system is loaded into memory.

What's the difference between a cold and a warm boot?

A cold boot is the opposite of a warm boot.

Cold booting is accomplished by turning the computer on and off, or on Apple //e and //c computers, by issuing the key sequence CONTROL RESET . Cold booting erases everything stored in RAM memory.

In a warm boot, the computer is not turned off and the information stored in RAM is not totally erased. To warm boot, use the PR#6 command.

Warm booting a DOS 3.3 slave disk (the kind produced by the INIT command) does not destroy the hi-res buffers and your picture will be intact. Warm booting may or may not reload the language card area of memory, depending on the disk being booted.

Warm booting ProDOS or DOS 3.3 master disks (the kind produced by the MASTER CREATE program on the DOS 3.3 System Master disk) will destroy hi-res memory.

If possible, we prefer warm booting to cold booting. Turning a computer on and off each time you switch software promotes wear and tear on the fragile power switch and chips. Commercial software, especially games, may not give you any choice—you may have to turn the computer on and off.

When can I insert and remove disks from the drive?

Because all but the earliest Apples try to boot DOS when they start up, you should insert your start-up disk prior to turning the computer on. You can remove it either before or after you've turned the computer off. On other makes of computers, these procedures may be different. If you use more than one type of computer, check the documentation that came with the system before you follow your Apple habits. Even on the Apple, there is one time when you may not remove or insert a disk.

You should only remove or insert a disk from the drive when the drive-in-use light is **not** lit. If you remove a disk when the drive is spinning, you risk removing it when data is being written to the disk. You'll scramble the information on the disk because the disk read/write head will spew data across the disk without regard to disk sectoring or catalog structure. Depending on the amount of scrambling, you may or may not be able to recover your data. Don't risk it.

A related problem is pressing CONTROL RESET when the drive is spinning. If data is being written to the disk, you'll lose it and possibly all the data on the disk. Sometimes you have no choice but to use CONTROL RESET when the drive is spinning because a program has crashed or something is drastically wrong. You do keep copies (backups) of all important data, don't you?

If you have an Apple II, IIplus, or //e, and try to start up with a disk that will not boot, the drive will spin forever. The best solution to this problem is to press CONTROL RESET to turn off the drive and try another disk. In this case, we know the computer was trying to **read** from the disk and the data on the disk will not be harmed by the reset procedure. On the Apple //c a nonbootable disk will bring up the message "Check Disk Drive" and the drive will stop spinning.

On other types of computers, these procedures may be different. Inserting or removing a disk at arbitrary times may destroy the information on the disk. If you use more than one type of computer, check the documentation that came with the system before you follow your Apple habits.

What's a prompt?

The prompt is a character or group of characters displayed by a program to indicate that it's waiting for you to type something.

The prompt depends on the program being run. The Apple system monitor uses an asterisk (*), Integer BASIC uses a greater-than sign (>), and Applesoft uses a right square bracket (]). The prompt alerts you as to which Apple system you're talking to.

Application programs can generate other prompts to indicate that they're waiting for input. Often the prompt will come in the form of a question. A BASIC program normally prints a question mark to indicate that it's waiting for input. Programmers can replace this question mark with a different prompt.

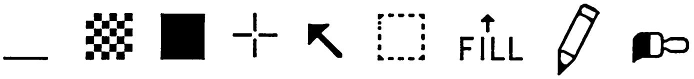
What's a cursor?

A cursor is a character displayed by a program to indicate your position on the screen.

On the text screen the cursor indicates where you are typing—when a character is typed, the cursor moves to the next typing position. The Apple system monitor is responsible for most Apple text cursors. Because the monitor has changed over the years, so has the cursor. Apple II and Apple IIplus computers produce a flashing rectangle as their cursors. Apple //e and //c computers produce a flashing checkerboard when they're in 40-column mode and a nonflashing rectangle when they're in 80-column active move. The system monitor deals with details such as what to do when the cursor hits the right edge of the screen.

Graphics programs use cursors to indicate where you are drawing on the screen. The most common graphics cursor is a cross-hair cursor with the point of drawing being the center of the cross-hairs.

Programmers can change the shape and behavior of the cursor. A flashing underscore is a common alternative cursor for the text screen. Graphics programs may use a variety of different cursors for different functions such as a tiny pair of scissors for cut and paste operations, a brush for painting, a hand for sliding the screen, and so on.



What’s a menu?

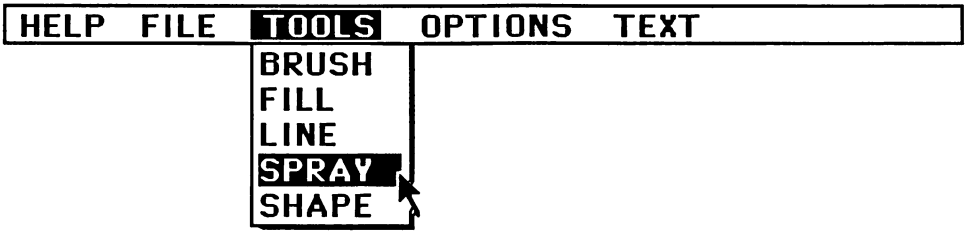
Menus are structures that present a list of options.

Programmers have always had to provide a means to allow users to control their programs. The most common method of controlling a program is through command keys. For example, a paint program might use CONTROL C as a clear screen command or CONTROL B to change brushes. The problem with command-driven programs is remembering the commands. Programs, especially powerful ones, may have 50 or more commands.

Menu-driven programs were invented to alleviate the mental overhead of remembering a large number of commands. In a menu-driven system, the user is presented with a screen of commands and prompted to select one of them. The selection methods vary from using the arrow keys to moving an inverse highlighted bar to pointing with the mouse or joystick.

Menus may be nested to simplify the screen display. For example, a paint program may offer a "change brush color" item on its main menu. Selecting this option produces a menu of colors. Too much nesting can make a menu-driven program more cumbersome than a command-driven one.

Like fashion, menu styles change with the times. The success of the Macintosh has popularized pull-down menus. In a pull-down menu, the main menu choices are shown on the screen, usually across the top. A selection is made by using the input device to move the cursor to the desired menu item and pressing the button on the device. Holding down the button while moving the cursor downward (much like pulling down a window shade) reveals another set of options. By releasing the button when the cursor highlights the desired option, the selection is made.



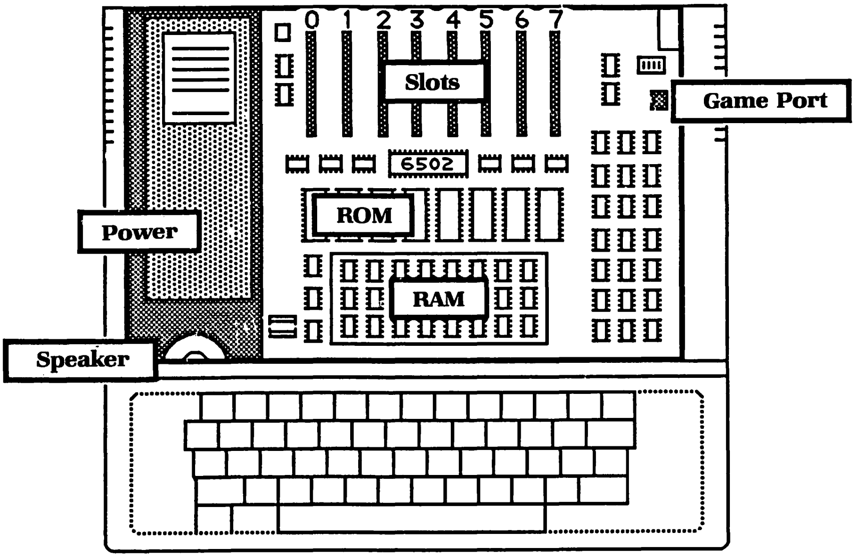
What's inside the computer?

If you use an Apple II, IIplus, or //e computer, chances are you've looked inside the computer. Apple //c computers can't be opened, but their innards look like a compacted Apple //e.

The main circuit board, a green plate covering most of the case bottom, is called the motherboard. The motherboard contains the main circuitry of the computer—fine wires running throughout the board connecting lots of small electronic components called chips. (See What's a chip? p. 31) The connecting wires between the chips are collectively termed the "bus" (or "buss"). The most important chip, and one of the largest, is the 6502 microprocessor, the "brains" of the Apple.

Memory chips are connected to the 6502 through the bus. Most Apples have 64K or 128K of memory. K stands for 1,024 in the world of computer metrics; 64K is 65,535 memory cells. This memory is volatile—when the power goes off, it's erased. Computists term volatile memory RAM (Random Access Memory). The Apple has another kind of memory which doesn't erase when the power is off and can't be easily changed. This nonvolatile memory is called ROM (Read Only Memory), (See What's memory? p. 32.)

All Apples have keyboards, a game port, a speaker, and video output. Apple II, IIplus, and //e computers have slots to connect I/O devices to the system bus. Apple //c computers have ports which, like slots, provide expanded I/O capabilities. (See What's a slot? p. 12 and What's a port? p. 13.)



What's a chip?

A chip is a silicon wafer about the size of your little fingernail. The part of the chip you see is the housing. Chip builders, using microscopes, encase the chip in this housing (usually plastic) and connect it to metallic legs or "pins." The pins are connected to the wiring on the motherboard.

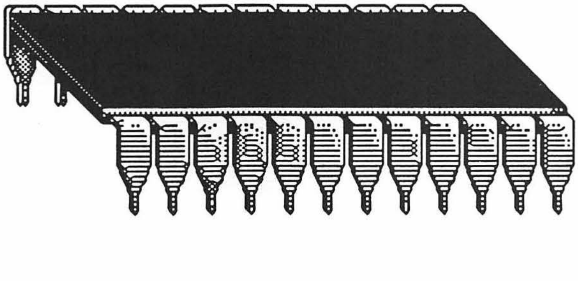
Each chip has a specific function in the computer: some are memory, some are gates to create the video display, one contains the character set you see on the CRT, and so on. The Apple //e and Apple //c contain significantly fewer chips than the Apple II and Apple IIplus computers—Very Large Scale Integration (VLSI) has enabled the function of many chips to be embedded in one.

Among these chips, one—and only one—makes the Apple a computer. Nearly every activity of the Apple is controlled by this large chip—the microprocessor. It has a name: the 6502. (The Apple //c and the enhanced //e computers have a 65C02 microprocessor, a newer and slightly more powerful version of the 6502.) A microprocessor is a type of central processing unit (CPU). The CPU computes. It can add numbers, subtract numbers, compare numbers, and so on. When the CPU instructions are sequenced into a program, astounding things can be done.

Memory chips enable the microprocessor to save the results of its computations and provide a place to put programs and other data. Without memory, the CPU could do very little. (See What's memory? p. 32.)

Some motherboard chips provide the Apple with input/output (I/O) capabilities. I/O gives us a way to tell the CPU what we want it to do. Other chips inside the Apple control how the video signal is constructed, how the 6502 manages its memory, and so on.

Add-on devices like mouse controller cards for the Apple II, IIplus, and //e also contain chips to control the functions of the device.



What's memory?

The central processing unit (CPU) is like an idiot savant. It can perform remarkably fast operations but it can't remember what it has done. Enter memory.

Memory provides the CPU with a place to store numbers that are used in computation and a place for the computation's results. Memory is also used to store programs that translate the CPU's operations into meaningful computation.

There are two kinds of memory: internal and external. Internal memory is wired directly to the microprocessor; external memory is located outside of the computer—on a floppy disk, cassette tape, hard disk, or other device. Internal memory is actively used by the CPU in the Apple; external memory is used as archival storage.

Internal memory is either RAM or ROM. RAM is an acronym for Random Access Memory and is a misnomer. Floppy disks, ROM, and many other computer storage devices are random access. What makes RAM special is that the information it stores can be retrieved or changed very fast—RAM is often referred to as Read/Write memory. This information is stored only temporarily—when the computer's power is turned off, RAM loses all it contains.

ROM is much like RAM except that ROM maintains the information it stores even though the power is off. Most ROM can't be easily changed—the information is said to be "burned into" the chip.

RAM is used to store user programs and data, information that changes each time the computer is used.

ROM is used to store programs to boot the disk, BASIC, and other system programs that do not change.

How much memory does an Apple have?

The Apple's CPU can only use 64K (or 65,635 bytes) of internal memory (both RAM and ROM). The first Apple II computers were sold with as little as 4K of memory, with an easy provision for upgrading to 48K. Soon Apple developed the Language Card, which enabled Apple II and Apple IIplus computers to have a full 64K of RAM. This RAM is in addition to the 12K of ROM—a clever switching arrangement assures that only 64K of this memory is available at one time.

The Apple //e has 64K RAM built in using the same switching arrangement in the Language Card.

For the Apple //e and the Apple //c, Apple extended the switching arrangement to encompass another full 64K of RAM (on the optional Extended 80-column card for the //e, and built into the Apple //c). This extra memory cannot be used by some programs—especially older programs for the Apple II and Apple IIplus computers—but does make several useful features available, including RAM disk capabilities (See What's a RAM disk, p. 66) and double hi-res graphics. It does not extend the memory directly available to BASIC programs.

Can more memory be added to the Apple?

Circuit boards can be plugged into the slots of the Apple II and Apple IIplus computers to extend the memory beyond 64K, but this extra memory can be used with only a handful of programs, and then only to store data. It cannot generate double hi-res graphics.

The Apple //e's RAM can be extended with an Extended 80-column card or the equivalent. This will add an additional 64K to the computer for double hi-res graphics and as extra storage for some programs. Manufacturers other than Apple sell memory expansion boards containing up to 1,000K (1M). This extra memory can be used as a RAM disk and a handful of programs can use it directly.

The Apple //c comes with 128K memory and double hi-res graphics capabilities. Officially, more memory cannot be added to the //c, but manufacturers other than Apple have created memory expansion circuitry that plugs into the //c motherboard and can provide large RAM disks or other specialized functions.

What's a page?

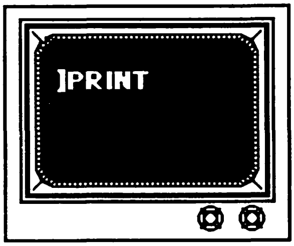
In standard use, a page refers to a 256 byte chunk of memory. There are 256 of these in the Apple's 64K of memory (256 × 256 = 65,535).

Pages are identified by the first two digits of their location in memory given in hexadecimal. (See What's binary, decimal and hexadecimal? p. 36.) Thus, the memory location 768 (or $0300 in hex) is the first location on page 3, location 784 (or $0310) is the sixteenth location on page 3, and location 8197 (or $2005) is the fifth location on page 32 (or $20). The 6502 microprocessor is constructed so that certain instructions can only be carried out within a page. In the Apple, many pages have preassigned functions. (See What's a memory map? p. 39.)

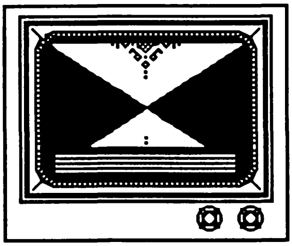
Graphics users frequently use the term "page" in another sense. Since we view the CRT in its entirety, rather than as separate microprocessor pages, the memory used to contain a screen is sometimes called a page. In this usage, the word "page" is identical to the graphics term "frame buffer." Using this parlance, the Apple has two text pages and two hi-res graphics pages.

Each text page is made up of four microprocessor pages; each hi-res graphics page is made up of 32 microprocessor pages.

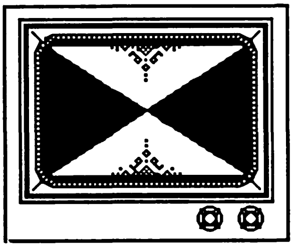
Because there's enough room in a 48K or 64K Apple to store a third hi-res graphic, you may sometimes hear of "graphics page 3." Unfortunately, there is no way to view a graphic on this page. It must be moved into one of the real graphics pages for viewing.



Text page



Hi-res page 1



Hi-res page 2

What are bytes and bits?

Computer memory (both ROM and RAM) is organized in bytes. There are 64K bytes (more accurately, 64 × 1,024, or 65,536 bytes) in some Apples and 128K bytes (or 131,072 bytes) in Apple //c and Apple //e computers equipped with extended 80-column cards.

Think of each byte as a mail slot in a vast post office. The bytes, like mail slots, are organized and numbered from 0 to 65,535. To read information from a byte, you specify the byte number. To store information into a byte, you specify the byte number and the information to be stored. (See What are PEEK, POKE, and CALL? p. 94.)

Like a mail slot, a byte can only store so much before it becomes full. A byte can store 256 different information codes. As far as the computer is concerned, these codes are numbers from 0 to 255, but we can choose to interpret these codes in many ways. The codes could represent letters or numbers, program statements, dots on the graphic screen, and so on. How a byte is interpreted by the computer depends on the program you're running.

Why can a byte hold only 256 different codes? The answer lies in how the codes are stored. Imagine a room with eight lamps. Depending on which lamps are lit and which are unlit, there are 256 possible combinations of lamp lighting. There are eight electronic switches inside each memory chip; each switch can be on or off. This gives 256 different combinations of switch positions from all off to all on and everything in between.

Each individual switch is termed a bit. Because a bit can store only two values, 1 and 0, the number system based on bits is called "binary" from the Greek prefix "bi" meaning "2."

Programmers use a 0 to represent a bit where the switch is off and a 1 to represent a bit where the switch is on. The location of a bit within a byte determines its contribution to the code stored in that byte. A byte with bit 0 on and all others off yields a code value of 1 and a byte with bit 7 on and all others off yields a code value of 128. Intermediate values are determined by adding up the values of each bit.

What are binary, decimal, and hexadecimal?

Binary, decimal and hexadecimal are three counting methods or number bases. You are already familiar with decimal, which you use all the time.

Binary is simply a direct representation of bits. Consider the following arrangement of bits:

▮▮▮▮▯▯▮▯ or 00001101

When expressing it as bits 00001101,we are counting in binary. (See What are bytes and bits? p. 35)

▮▮▮▮▯▯▮▯ equals 1 + 4 + 8 = 13

When expressing it as the number 13, we are using our normal counting method, decimal. Each bit contributes to the decimal representation as follows:

▮▮▮▮▮▮▮▯	1	▮▮▮▯▮▮▮▮	16
▮▮▮▮▮▮▯▮	2	▮▮▯▮▮▮▮▮	32
▮▮▮▮▮▯▮▮	4	▮▯▮▮▮▮▮▮	64
▮▮▮▮▯▮▮▮	8	▯▮▮▮▮▮▮▮	128 (the hi-bit)

Hexadecimal, or just hex, is another method of counting. As the name implies, this is counting in sixteens—hex means 6 and decimal means 10. In decimal we use ten digits (0 to 9) to represent numbers, but in hex we use sixteen digits. In decimal, when we run out of digits, we move over one place: 8, 9, **10**, **11**, and so on . In hex we do the same thing, but after we've counted to sixteen. The letters A through F provide the extra digits to represent the decimal numbers from 11 to 15. Therefore, 8, 9, 10, 11, 12, 13, 14, 15, 16 in decimal becomes $8, $9, $A, $B, $C, $D, $E, $F, $10 in hexadecimal. (By convention, hex numbers are prefixed by a dollar sign.) In hex, 13 is equal to $D.

▮▮▮▮▯▯▮▯ equals $D

The prime reason for counting in hex is simplicity when specifying a memory location. Memory locations are numbered from 0 to 65,535 in decimal, which translates to $0000 to $FFFF in hex. It's simpler to remember $4000 than 16384.

How do I convert binary numbers to hexadecimal?

Converting from binary to hex is simple. Each byte can be represented by two hex digits; each hex digit, then, represents a half a byte. A half a byte has a special name: a nibble (or nybble).

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

Translating a byte to hex is a matter of putting the correct two hex digits together. Therefore, 00011010 becomes $1A, 10110011 becomes $B3, 11100101 becomes $E5, and so on.

All memory locations in the main 64K memory of the Apple can be specified in two bytes. $FFCB, $4020, $0300, and $0802 are memory locations in hex. In this book, we give both the decimal and the hex representation of all memory locations where you may use either in a command. If you are only allowed to use decimal or hexadecimal representations, we specify the correct represention.

The Appendix (p. 283) contains a list of the important memory locations in both decimal and hexadecimal forms.

How do I convert hexadecimal numbers to decimal?

If you're working in BASIC, you must convert hex numbers to decimal before using them. (See What are binary, decimal, and hexadecimal? p. 36) Most of the numbers to be converted to decimal will be either two hex digits long, representing one byte's worth of information, or four hex digits long, representing an address in memory.

The first step in conversion is to translate the hex digit into a decimal number. The numbers $A through $F represent decimal numbers from 10 to 15.

Next, each decimal number is multiplied by the correct hex place value. The place values in a four-digit hex number are [4096] [256] [16] [1]. For comparison, the place values of a four-digit decimal number are [1000] [100] [10] [1].

Finally, the results of this multiplication are added together.

It sounds more complicated than it is. Here are two examples:

Number for conversion:	$03D0			
Separate into digits:	$0	$3	$D	$0
Translate digits to dec:	0	3	13	0
Multiply by place value:	4096	256	16	1
	0	768	208	0
Add together:	976			

Number for conversion:	$C057			
Separate into digits:	$C	$0	$5	$7
Translate digits to dec:	12	0	5	7
Multiply by place value:	4096	256	16	1
	49152	0	80	7
Add together:	49329			

BASIC can do most of the conversion for you.

Key in : PRINT 3*256+13*16 RETURN
BASIC will display the answer: 976
Key in: PRINT 12*4096+5*16+7 RETURN
BASIC will display the answer:49329

BASIC multiplies before it adds so there's no need for parentheses or other symbols.

What's a memory map?

A memory map is a diagram which shows how the Apple uses memory. Here we show a map of the typical 64K system.

Starting at the bottom of memory, you'll note that the first three pages of memory, $0, $1, and $2, are devoted to system operations. Zero Page (addresses 0 to 255) and page 1 (addresses 256 to 511) are used extensively by the 6502 microprocessor. Page 2 is used as an input buffer—everything typed on the keyboard is temporarily stored here. Page 3 is mostly free and is used by many graphics programmers to store small shape tables, picture unpacking routines, and so on. The upper portion of this page is used by DOS.

Continuing upward in memory, the next four pages (locations 1024 to 2047) are devoted to the text screen. That's right, the text screen. The text screen is actually a snapshot of memory. This type of display is termed a "memory mapped display." Anything stored on these pages appears on the text screen.

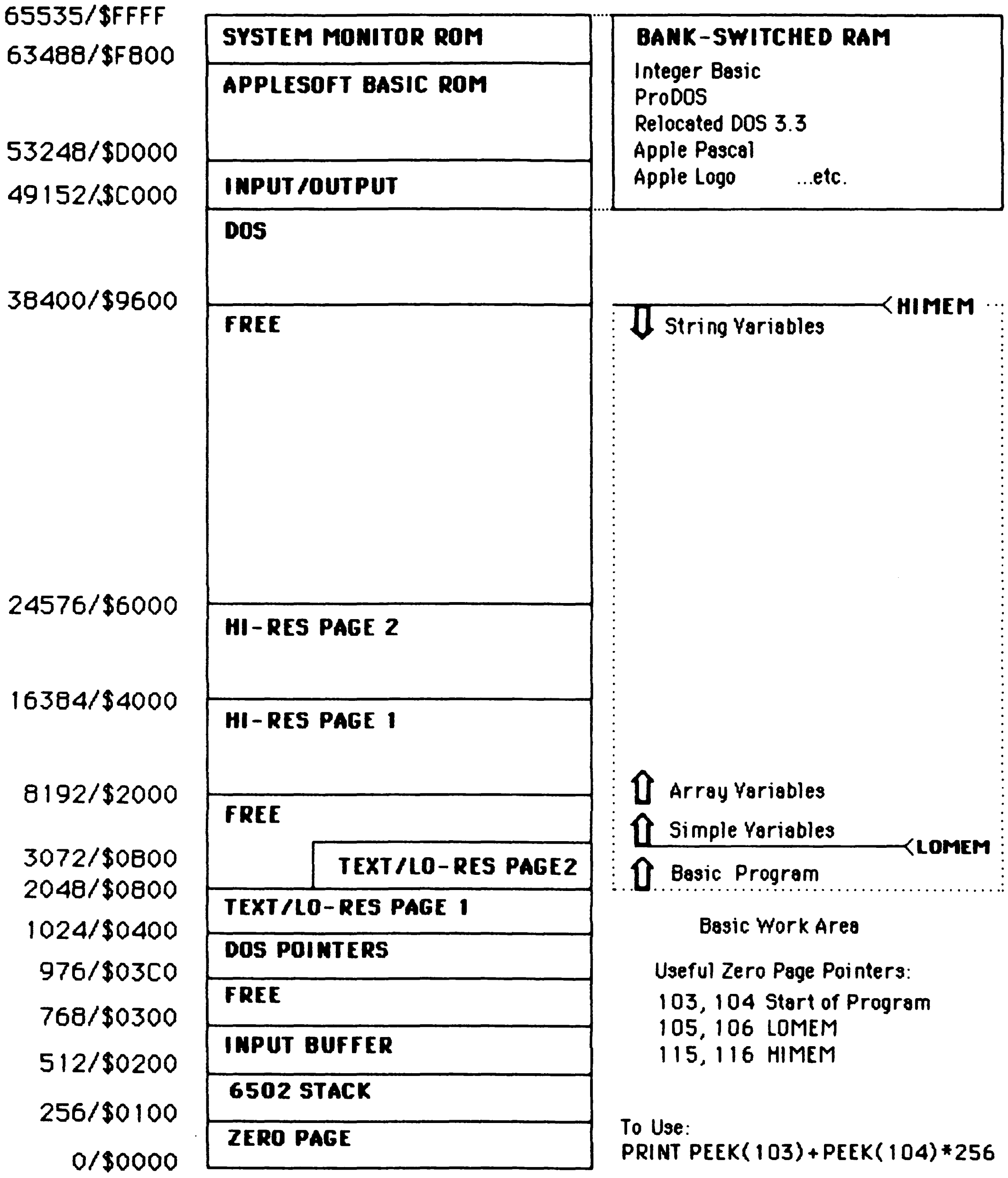
Next, the largest portion of the Apple's memory (addresses 2048 to 38400) is devoted to BASIC for storage of programs and variables. Notice that this area of memory conflicts with text/lo-res page 2 and both hi-res graphics pages. This conflict is one of the prime purposes of a memory map. Beginners rarely have problems with conflicting usage of memory, but as your programs and graphics become more sophisticated, you'll need to manage memory, rearranging how the Apple uses this block of memory. (See How do I relocate programs? p. 102.)

Continuing upward, DOS occupies the next segment of memory, although some commercial DOS utilities may allow you to move DOS onto the Language Card or bank-switched memory. At this writing, ProDOS cannot be moved.

Next, a section of addresses is devoted to I/O. This area contains the soft-switches that enable you to switch between different display modes, ROM routines to manage an 80-column card, printer, or mouse, and so on.

At the very top of memory, we find the Applesoft BASIC and System Monitor ROM. The same space (on 64K or 128K Apples) also contains RAM. Programmers can switch between the two, effectively adding an additional 16K to the Apple's memory. Note that BASIC cannot be active when the switch occurs; because BASIC is in the ROMs, a switch would make the Apple forget BASIC!

Memory Map—Apple II Computer with 64K



Useful Zero Page Pointers:

103, 104	Start of Program
105, 106	LOMEM
115, 116	HIMEM

To Use:

```
PRINT PEEK(103)+PEEK(104)*256
```

What's the system monitor?

The Apple system monitor is a program stored in ROM that will perform the Apple's fundamental processes: accept input from the keyboard, put a character on the screen, and so on. The system monitor is written in machine language and is used by BASIC, DOS, and most assembly language programs.

As the hardware became more and more sophisticated with each generation of Apples, the system monitor changed. The monitor of the original Apple II is 8K long, as is the Autostart monitor found in the Apple IIplus and most Apple IIs. The primary difference between these is that the assembly language support of the original monitor was replaced by support for autoboot from disk and enhanced screen editing in the Autostart monitor.

For the Apple //e, the monitor was stretched to 12K of code, primarily to support the 80-column card.

For the Apple //c and the enhanced Apple //e's, the system monitor is 16K in length and includes support for microprocessor interrupts—critical for mouse and modem operations.

Because Apple carefully documented the monitor from the start and told programmers how to use it, most programs that ran with the original monitor run fine under the current one. The converse may not be true if the program uses interrupts or other special capablities of the current Apple system monitor. This compatablity situation is termed "upward compatiblity."

A listing of the system monitor that comes with the Apple can be found in the technical reference manual for that system.

You can enter the monitor and examine memory locations, move data from one address to another, and perform other memory manipulations. Programmers can use subroutines in the monitor to make their programs shorter and more efficient.

For the most part, BASIC is sufficient for beginning and intermediate programmers.

How do I use the Apple system monitor?

The Apple system monitor is capable of many functions that make a programmer's life easier. Beginners need not be concerned with most of these functions, but there are a couple of monitor instructions that everyone benefits from mastering.

Enter the monitor by typing CALL -151 RETURN . The normal Applesoft prompt, the right bracket, will be replaced by the monitor's asterisk (*) prompt. The Apple will no longer respond to Applesoft commands, but DOS 3.3 and ProDOS will function as expected.

Only a small set of commands are legal for the monitor—all others will produce a beep from the Apple's speaker.

To list the contents of an area of memory, specify its starting and ending address, separated by periods. The monitor, unlike BASIC, expects addresses expressed in hexadecimal. Key in 300.310 RETURN and the monitor will dutifully present you with 17 numbers. ($10 is 16, and there are 17 numbers from 0 to 16.) Depending on what you were doing with the computer before you entered the monitor, these 17 numbers could be almost anything.

There's another form of listing, called a disassembly, that often proves useful. To generate this type of listing, key 300L RETURN and you'll see 21 lines of information. The first four columns represent the addresses being listed followed by a dash. The next one, two, or three columns present the hexadecimal contents of the memory locations. The final group of columns represent the assembly language translation of the contents of the memory locations. It'll look something like this (but your numbers may be different):

```
*300L
0300- AD 30 C0     LDA $C030
0303- 88           DEY
0304- D0 04        BNE $030A
0306- C6 FF        DEC $FF
0308- F0 08        BEQ $0312     ...and so on
```

The primary use of a disassembly is for proofreading assembly language programs that you find in magazines or books. The assembly language should match the listing in the magazine or book; if not there's an error.

Another monitor command that's useful to beginners is the monitor store command. You can use this command to store graphics data such as a shape table or a machine language program from a book or magazine. The command is a colon. Enter an address, followed by a colon, followed by the values to be stored in the address specified. Try the following:

```
300:AD 30 C0 88 D0 04 C6 FF RETURN
:F0 08 CA D0 F6 A5 FE 4C 00 03 60 RETURN
FE:FF FF RETURN
```

Proofread your typing; if you've made any errors start again from the top. When you're sure it's correct, type 300G RETURN . You should hear a sound from the speaker. The first two lines stored an assembly language program into locations $300 (768 in decimal) to $312. Note that the colon without a preceding address denotes a continuation from wherever it left off with the previous store command. The next line stored data in locations $FE (254) and $FF (255), and the final line executed the program with the monitor go (G) command.

It's dangerous to use the G command unless you're sure the program has no errors—unlike BASIC, the monitor has few protections against destroying DOS or other critical parts of memory. (By the way, this program is the same program POKEd in Music Bonus: Music Routine p. 46)

To return to BASIC, type CONTROL C RETURN or CONTROL B RETURN . The former will not erase your BASIC program; the latter will.

Because assembly language subroutines and data are so important to graphic programming, you should become comfortable entering and using the monitor list and store commands.

What's a crash?

A computer system is said to crash when it becomes inoperative. Suddenly everything goes haywire, or stops completely, and you're powerless to do anything short of turning off the system.

You can cause the computer system to crash, and lose your work, if you make serious errors in programming or if the program you're using is not adequately protected against bad input.

As a precaution, keep back up copies of all important programs and data. As you work, save your data every twenty minutes or so. When working on a graphic, we recommend saving it as soon as you have done something to it that you know you like—a choice brush stroke is hard to recreate.

What do I do if my system crashes?

First, make sure that the system really has crashed. Sometimes the system appears dead but can be revived with a CONTROL RESET . For example, if you try to print to a nonexistent printer, the Apple will be totally unresponsive. CONTROL RESET will return control to you. In most cases, this key sequence will leave your graphics or program intact.

If DOS or ProDOS is still usable—see if you can CATALOG the disk—you can SAVE your program or BSAVE your graphic.

If DOS or ProDOS no longer work, try typing CALL 976 RETURN and you may reactivate DOS.

CONTROL RESET causes some software to reboot the disk and you will lose your BASIC program that was in memory when the system crashed. You may still recover you graphic, if you switch disks. (See How Can I save my pictures when the system crashes? p. 163)

If CONTROL RESET does not revive the computer, you have no alternative but to turn the Apple off and back on, losing the contents of memory.

What makes the Apple beep?

By pressing CONTROL G , you can make the Apple beep. To put beeps into a BASIC program, print the ASCII bell character, PRINT CHR$(7) RETURN .

All Apples have a built-in speaker that can be "tweaked" to produce a tiny buzz. A beep or other sound effect can be effected by tweaking it rapidly. The speaker on the Apple II is controlled by the I/O port at location -16336 (49200 or $C030).

From Applesoft, you can click the speaker by entering POKE -16336,0 RETURN . The click you hear will be very soft. Clicking the speaker rapidly in succession produces a variety of sound effects. Try this: FOR I = 1 TO 10: POKE -16336,0: NEXT I.

To produce higher frequency tones, the speaker has to be toggled faster, but BASIC is too slow. Higher frequency tones and music require a small machine language routine. (See Music Bonus: Music Routine. p. 46)

Don't get carried away. Beeps and sound effects can be very annoying and only the Apple //c has a volume control knob.

Can the Apple speak?

The Apple's speaker can produce speech or sounds resembling speech, but whether male or female, it will have a definite "electronic" sound to it.

There are several commercial programs available that let the Apple talk, but when the Apple is speaking, nothing else can happen: the CPU is too busy talking. You can't have the Apple speak while a figure is animating; you'll have to design pauses in the action for the dialogue.

If speech is a necessary part of your graphic design, you can purchase an external speech synthesizer that plugs into the slots of the Apple II, IIplus, or //e, or the ports of the //c. Several different manufacturers produce such peripherals, and most of them offer music and sound effects in addition to speech. Generally, finer quality speech is produced with these devices than with software.

Music Bonus: Music Routine

When tweaked from BASIC, the Apple speaker produces only clicks and buzzes, but from machine language the speaker can produce music.

```
10 FOR I = 0 TO 18: READ T
20 POKE 768+I,T: CS=CS+T
30 NEXT I
35 IF CS<>2713 THEN PRINT "ERROR IN DATA": STOP
40 FOR P = 10 TO 250 STEP 10
50 POKE 254,P: POKE 255,50: CALL 768
60 NEXT P
100 DATA 173,48,192,136,208
110 DATA 4,198,255,240,8
120 DATA 202,208,246,166,254
130 DATA 76,0,3,96
```

Lines 10 through 30 read the DATA and POKE it into memory starting at location 768. These DATA are a machine language subroutine. Be careful typing the DATA statements: one incorrect number could cause the computer to crash. We've included a check—the variable CS is used to accumulate a total value of the DATA statements. If CS does not equal 2713, the program will stop. This method is not foolproof because two typing errors could cancel each other out. SAVE the program before you RUN it just in case.

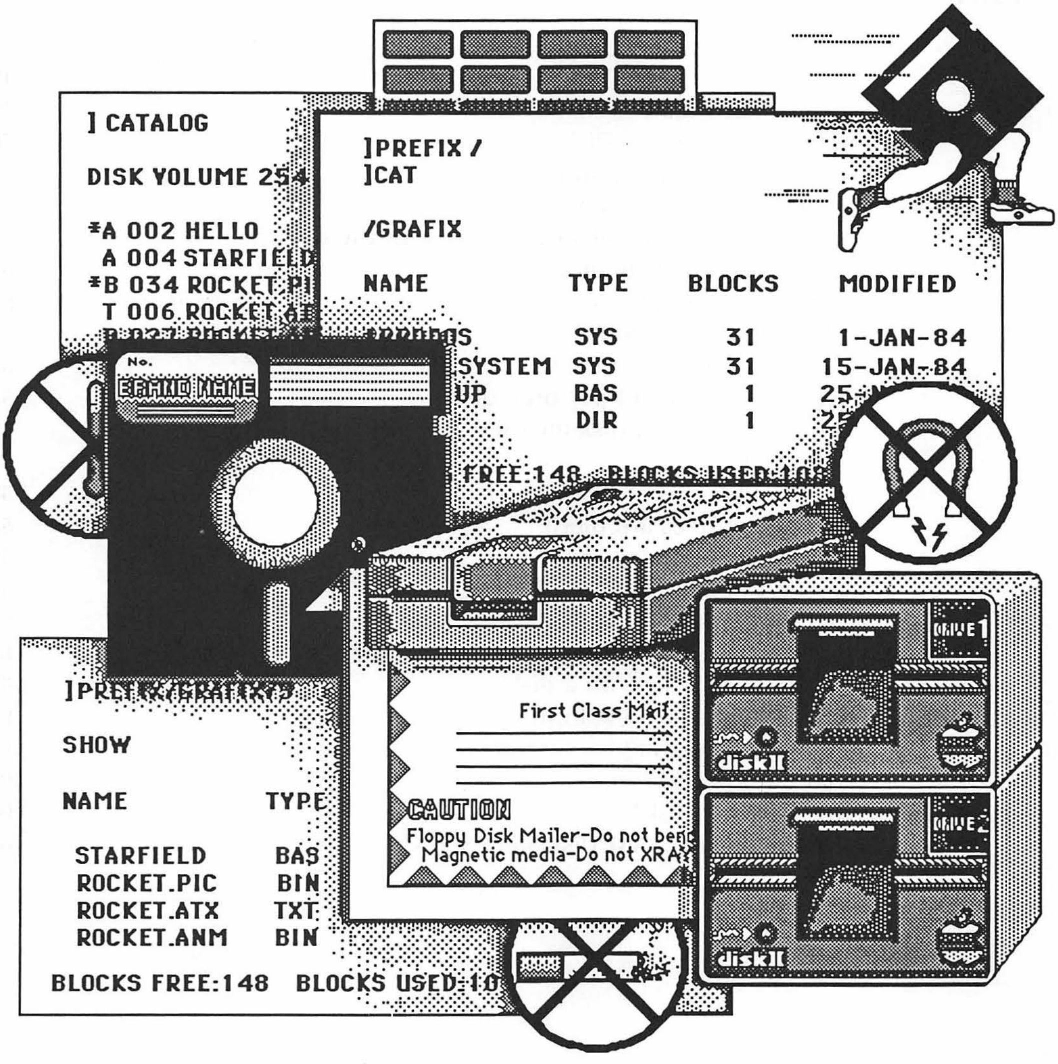
After the machine language subroutine has been POKEd into memory, the loop at lines 40 through 60 POKEs successive values into locations 254 (pitch) and 255 (duration) and CALL the music subroutine at 768. By changing these POKEs, you can create music and sound effects to accompany your graphics.

If you'd prefer to have the music routine as a binary file on your disk so that these program lines do not have to be in each of your programs, type the following command after verifying that the routine does work: BSAVE MUSIC.BIN,A768,L19.

To use the MUSIC.BIN file from your programs, put this line in your program: PRINT CHR$(4);"BLOAD MUSIC.BIN".

The MUSIC.BIN file must be on the disk with your program. After POKEing values for pitch and duration, CALL 768 will play a tone.

For more sophisticated music, consider a commercial music program such as Electric Duet by Insoft, or a music peripheral such as the Mockingboard from Sweet Micro Systems.



```
] CATALOG

DISK VOLUME 254

*A 002 HELLO
 A 004 STARFIELD
*B 034 ROCKET.PI
 T 006 ROCKET.AT
```

```
]PREFIX /
]CAT

/GRAFIX

NAME             TYPE    BLOCKS    MODIFIED

            S    SYS         31     1-JAN-84
      SYSTEM     SYS         31    15-JAN-84
          UP     BAS          1    25-
                 DIR          1    25

 FREE:148  BLOCKS USED:10
```

```
]PREFIX/GRAFIX/

SHOW

NAME            TYPE

 STARFIELD      BAS
 ROCKET.PIC     BIN
 ROCKET.ATX     TXT
 ROCKET.ANM     BIN

BLOCKS FREE:148   BLOCKS USED:10
```

About Disks and Files

About disks and files

What's a disk?

Without a medium on which to permanently store data—pictures, programs, shape tables, and so on—using computers would be tedious indeed. Data would have to be hand entered each time the computer is turned on.

Apple II, Apple IIplus, and Apple //e computers have cassette tape interfaces that allow you to store data on standard audio cassettes. Cassettes, however, are slow and unreliable.

Floppy disks were a relatively new medium when personal computers were invented and the Apple was the first "appliance" computer to offer disks and disk drives as an alternative to the cassette.

The disk itself is a ring of mylar coated with a magnetic surface. This surface is like the tape in a cassette tape. The disk is packaged in a case of flexible plastic which is lined with a material that cleans the disk as it spins in the disk drive.

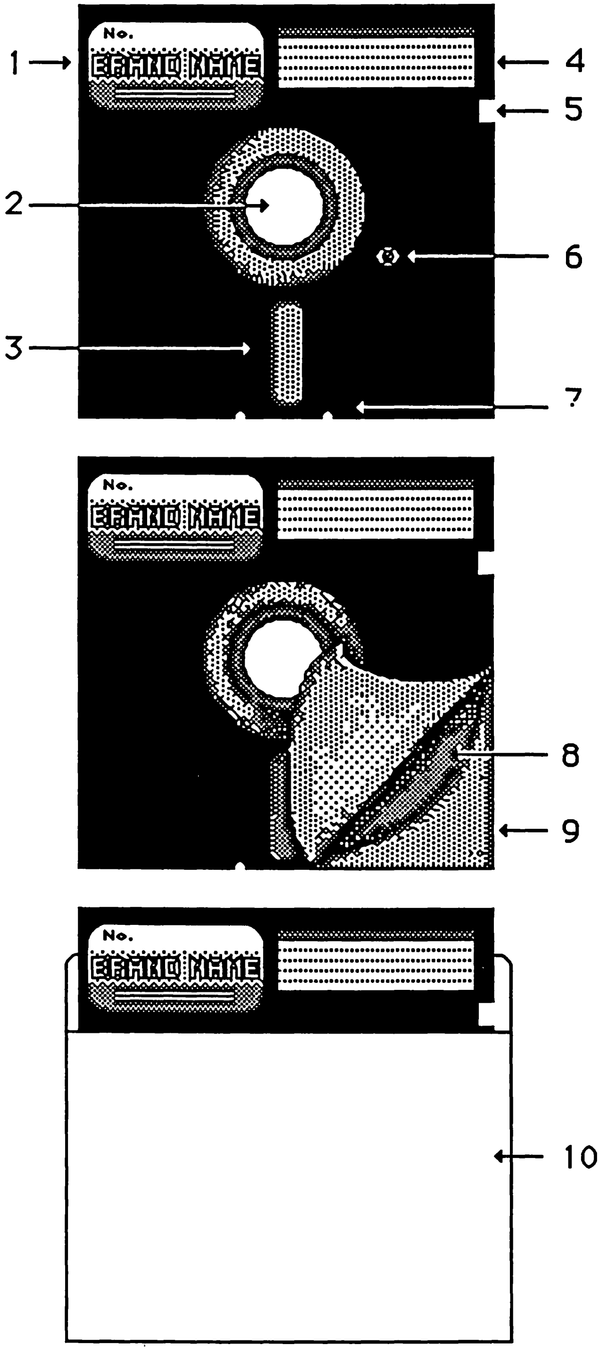
Information is recorded on the disk in concentric circles called tracks. Each track is divided into sectors. Apple disks record 35 tracks containing 16 sectors in each track. (The original Disk II used 13 sectors per track.) Each sector records 256 bytes of information, making the total capacity of the Disk II 35 × 16 × 256 or 143,360 bytes (140K) of information.

What's a hard disk?

Hard or fixed disks store much more information (and are much more expensive) than floppy disks. A hard disk has rigid plates coated with magnetic material. Because the disk is not flexible, information can be stored more densely than on floppy disks. A 5¼" Apple floppy disk stores 143K of data; the Apple ProFile hard disk can store 5,000K (5 Megabytes) or 10,000K (10M).

Because you cannot remove the disks on most hard disk systems, you must copy the information to floppy disks for backup. Then if the hard disk crashes, that is, if the disk becomes unreadable, you can restore the data from the floppies. It takes 35 floppies to backup a 5M ProFile and 70 floppies to backup a 10M ProFile!

Anatomy of a floppy disk



1 Manufacturer's label with info on type of disk and product I.D.*

2 Hub or center hole by which the disk drive rotates the disk.

3 Head window where read-write head of disk drive comes into contact with the disk.

4 Your label for recording disk file information.

5 Write-enable notch.

6 Index window where light beam in drive checks disk position and rotation - Not used on Apple II computers.

7 Relief notches used to position the disk.

8 Magnetic disk stores data. Spins inside a protective lined jacket.

9 Liner cushions and cleans the disk as it spins.

10 Protective disk jacket.

What kind of disks should I use?

The type of disk you use is determined by the type and capacity of the disk drive you're using. Disk capacity is determined by the number of sides used (one or two) and the track density which is measured in tracks per inch (48 tpi or 96 tpi). Use the disk type recommended in the documentation that came with the drive.

The Apple Disk II, the most common drive connected to Apples, holds 143K of information on a single side of one 5¼″ floppy disk. If you use this disk drive, you need single-sided, double-density (48 tpi) disks. The internal drive in the Apple //c and the drives in the Apple Duodisk are disguised Disk II's.

Double-sided drives have two magnetic read/write heads, and DOS knows whether to store a file on side one, side two, or even on both sides by splitting the file. Double-sided drives store upwards of 260K of information on a single disk and require double-sided, double-density (48 tpi) disks. Currently, Apple does not sell a double-sided drive for the Apple II family of computers, but independent suppliers do.

Quad density drives store four times more information than single density drives and require quad density (96 tpi) diskettes.

It does no harm (except to your budget) to use a higher quality disk than a drive is rated for, but the reverse is not true. Quad density drives may not properly format a single or double density diskette.

If Apple makes the Macintosh-style 3½″ drives available for the Apple II, you will need single or double-sided 3½″, double density disks. These disks store 400 or 800K respectively.

How many disks do I need?

Lots. Graphics use a lot of disk space and you'll want a make a copy of every disk. Only 14 hi-res images on a disk and a backup . . . disks, disks and more disks!

What are those little tabs that came with my disks?

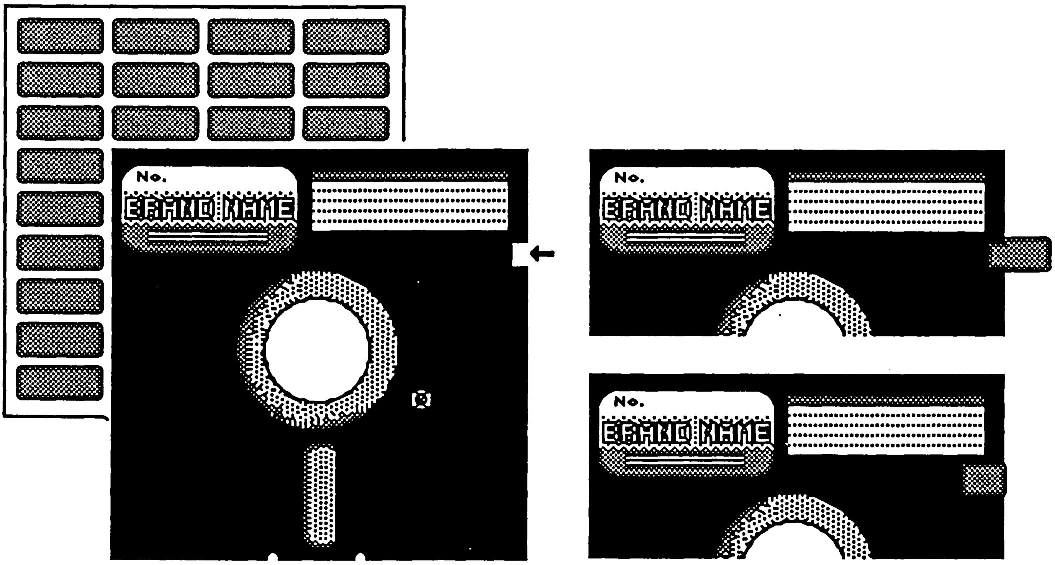
The tabs are called write-protect tabs.

On the upper right side of the floppy diskette there's a small square cutout called a write enable notch. This notch allows (enables) information to be written to the diskette.

Some diskettes don't have this notch, so there is no possibility of accidently writing to them and possibly destroying the data they already contain. The DOS 3.3 System Master and other Apple master disks are protected in this way.

As long as a disk has this notch, and the notch is not covered, the disk can be written to. When you want the disk protected, you must cover the notch; the write protect tabs are provided for this purpose.

Generally speaking, you should put a write-protect tab on all commercial software programs to protect them from being overwritten. Fold the tab around the notch, covering it completely. Be sure to press it down firmly or it will lift up and prevent you from inserting the disk into the disk drive. Your data disks must not be write-protected because you want to save your programs and pictures to them. When you've filled a disk with pictures and other data, you can put a tab on it.



How do I care for my disks?

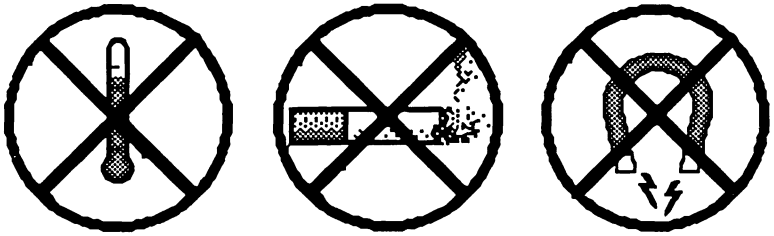
Disks are magnetic media, but unlike most magnetic media we use, the information is stored digitally. The loss of a single bit can affect a disk reading so that the disk is unusable. Disks must be treated with care.

Don't expose them to anything that can erase magnetic signals. Magnets are hidden everywhere: in CRT's, in telephones, in motors, and so on. Keep your disks away from these devices.

Extremes of temperature will cause a disk to physically shrink or expand and thus damage the disk. Dirt, dust, and smoke are not good for a disk's health. Oils from your fingers—or from a peanut butter sandwich—can cause dirt to adhere to the surface and obliterate information. Bending a disk can cause the surface to flake.

When using a disk, don't touch its surface. When not in use, keep the disk in the protective envelope it came in. Always store disks upright in their boxes. When you've accumulated a lot of disks they can be stored in specially designed disk cases. Use a soft felt tip pen when writing on the disk label. Always make back-up copies and store the backups away from the originals.

With proper care, a disk can live a long, productive life.



Do disks wear out?

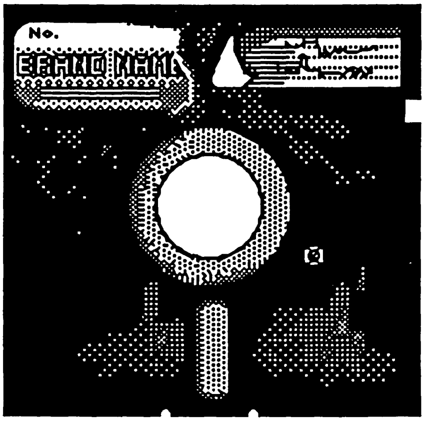
Yes. Disks have a life expectancy although we're not sure just how long it is.

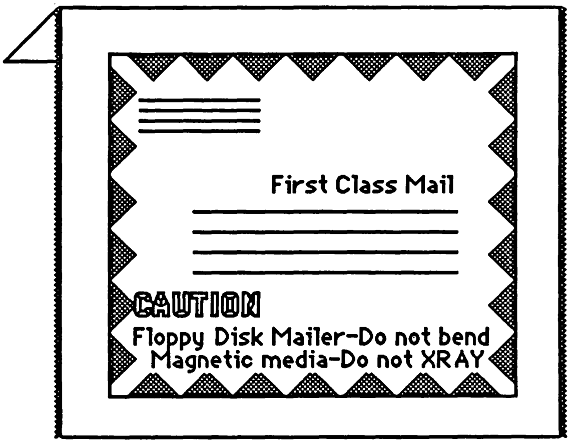
The disk only experiences wear and tear when you're actually reading or writing to it. Some disk manufacturers claim that their disks have completed over 10 million passes through the disk drive with no sign of wear. If a disk begins to act strangely—give frequent I/O errors, load or save incorrect data, and so on—copy the files to another disk and discard the old disk.

Is it safe to mail a disk?

Yes, as long as it's packaged properly. Commercial disk mailers can be purchased in stationary and computer supply stores. You can make your own mailer by putting the disk between pieces of rigid cardboard in a large envelope.

Always write the following in a conspicuous place on the envelope: "Caution: Magnetic Media: Do Not X-Ray. Do Not Bend."





What's on a disk?

Disk sectors contain magnetically recorded data. Apple uses a complicated encoding scheme to cram more information than should be possible on the Disk II mechanism. DOS and the disk drive take care of encoding and decoding raw data on a disk.

The information on a disk can be divided into three major groups: boot, catalog, and user data. This information is stored in files.

Boot information, which enables the disk to boot, is always recorded on track 0, sector 0 of the disk. On a normal, non-copy protected boot DOS 3.3 disk, DOS itself is stored on track 0, track 1, and part of track 2. The file containing DOS does not appear on the disk's catalog. On a ProDOS disk, ProDOS is a regular file on the disk and the ProDOS boot code finds and reads the ProDOS file. Copy protected disks must have normal, readable information on track 0, sector 0, but after that, anything goes.

Both DOS 3.3 and ProDOS store catalog information on the disk. The disk directory or catalog specifies, for each file, a name, a type, and where (in tracks and sectors) the file information is stored. DOS 3.3 and ProDOS store their directories on different tracks of the disk and use a different format for the data. Without a special program—like the file convert utility on the ProDOS User's Disk—ProDOS cannot read a DOS 3.3 directory and vice versa.

Encoded user data (your files) occupies the rest of the disk.

How many files can I put on a disk?

The maximum number of files is determined by the DOS you are using.

DOS 3.3 allows a maximum of 105 file names, whereas ProDOS allows 51 files per directory. In normal usage, most files are longer than the minimum size so you'll probably run out of disk space long before the catalogs are filled.

Graphics files take up exceptionally large amounts of disk space. (See How many pictures will fit on a disk? p. 161)

How do I prepare a disk for use?

The process of preparing a disk to store pictures and programs is called "initialization" and is different for DOS 3.3 and ProDOS.

DOS 3.3

Boot the computer with the Apple System Master diskette or another bootable DOS 3.3 disk. When you see the Applesoft prompt (]), type:

```
NEW RETURN
10 HOME RETURN
INIT HELLO RETURN
```

The disk-drive will whirr and chug as DOS writes formatting information onto the disk, erasing any information that might have been there. Never INIT a disk which has files you want to keep—they'll be destroyed.

The DOS INIT command formats a disk and saves a copy of itself to the disk. The INIT command also saves the resident BASIC program under the name that follows INIT. Tradition dictates that this first program be named HELLO, but any legal DOS 3.3 name is acceptable.

ProDOS

The INIT command is not available in ProDOS. You must use the ProDOS User's Disk's initialization program, which is accessed through the Volume selection from the main menu of the User's Disk. You'll be asked to specify the disk name as a part of the initialization process. The name must start with a letter preceeded by a slash, be fifteen letters or less long, and contain no spaces or punctuation other than a period. Examples of valid names are "/PIX1", "/MY.PICS", and "/GRAPHICS".

ProDOS does not create bootable disks. (See What's a data disk? p. 61) The initialization program doesn't save a copy of ProDOS to the new disk, nor does it save a HELLO program. To make a bootable disk, you must return to the ProDOS User's Disk and copy "PRODOS" and "BASIC.SYSTEM" to your disk using the File commands selection from the ProDOS main menu. To make a program run on boot, like the DOS 3.3 HELLO program, name your program STARTUP and ProDOS will automatically RUN, BRUN, or EXEC the program when the disk is booted.

What's DOS?

DOS, which rhymes with "boss," is an acronym for Disk Operating System. It's a program, written in machine language, that provides the Apple with the ability to read and write to a disk.

Apple has released several versions of DOS; each new release is an enhancement of the previous version. Sometimes the current release can't read or write disks created with the earlier version. Such is the case with DOS 3.3 and ProDOS. Technically, ProDOS is a far superior DOS, but there's a large quantity of DOS 3.3 graphics software. Chances are you'll be using both operating systems.

Although ProDOS can't read a DOS 3.3 disk, data files can be transferred between ProDOS and DOS 3.3 disks by using the utilities provided on the ProDOS User's Disk or Apple //c System Utilities disk. Some programs, those that use features of DOS 3.3 that aren't included in ProDOS, will not work under ProDOS, but picture and data files will be usable.

You use DOS—either one—by issuing valid DOS commands in immediate mode (at the Applesoft prompt) or in programs. (See How do I use DOS in a program? p. 93)

What's a file?

A file is a chunk of information. Although a file can be stored anywhere, most of the time files are on a disk.

What types of files are there?

Files differ in the type of information they contain. Both DOS and ProDOS can distinquish these differences. The disk catalog displays file types. (See Anatomy of a catalog. p. 59)

In DOS 3.3, the file type is indicated by a single letter in the second column of the disk's catalog.

Most of your work will be with Applesoft and Binary files. Applesoft files are created by the BASIC SAVE command and are executable Applesoft programs. Binary files can be almost anything—a full-screen graphic image, a shape table, a machine language program, an animation sequence, and so on. Binary files are created with the BSAVE command.

Some commercial software may create text files. A text file is a collection of characters stored in standard ASCII code. Text files are often used for data from word processing programs and are also used to save information a program needs, such as what brand of printer you have and which slot it's in. The program stores this information in a text file and reads it when needed so that you don't have to enter the same information each time you use the program.

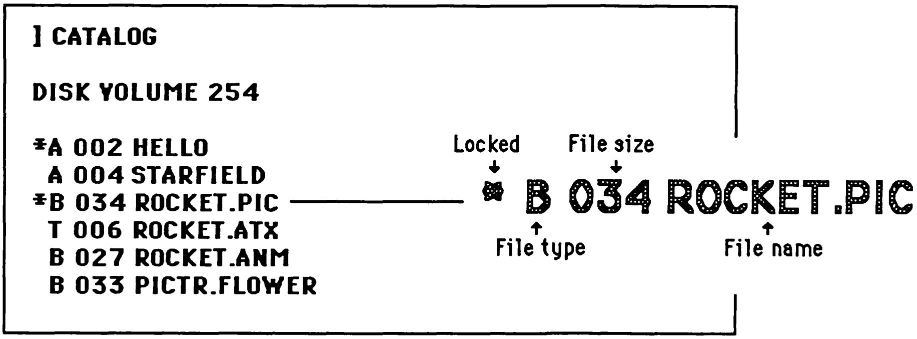
The other common type of file is an Integer BASIC file. Like Applesoft files, Integer files are programs stored with the SAVE command however these programs are in Integer BASIC rather than in Applesoft BASIC.

Under ProDOS, the file type is indicated by a three letter abbreviation in the "type" column of the catalog.

ProDOS recognizes all the file types recognized by DOS 3.3 and more. The most important file type under ProDOS is a system file. A system file is a special type of machine language program which connects ProDOS to a language or program being run. Without the file BASIC.SYSTEM or its equivalent, you can't program in Applesoft under ProDOS. Other types of ProDOS files include Pascal, AppleWorks, and variable tables.

Anatomy of a catalog

DOS 3.3 CATALOG

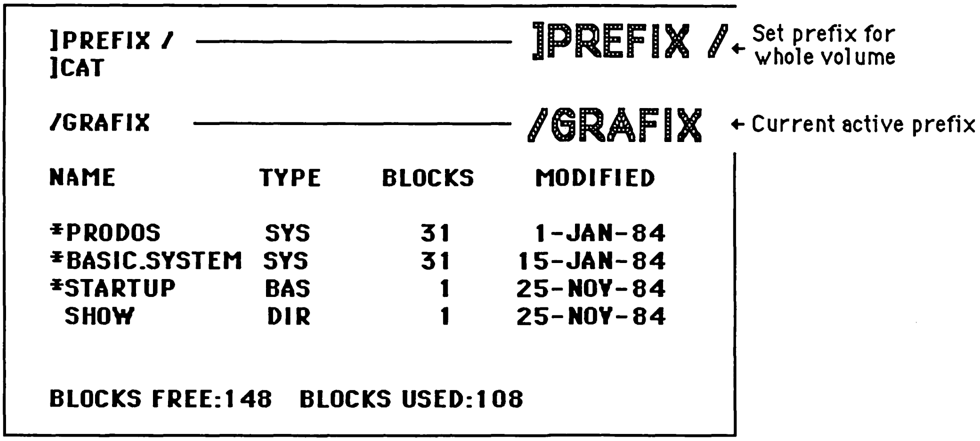


```
] CATALOG

DISK VOLUME 254

*A 002 HELLO
 A 004 STARFIELD
*B 034 ROCKET.PIC
 T 006 ROCKET.ATX
 B 027 ROCKET.ANM
 B 033 PICTR.FLOWER
```

ProDOS CAT



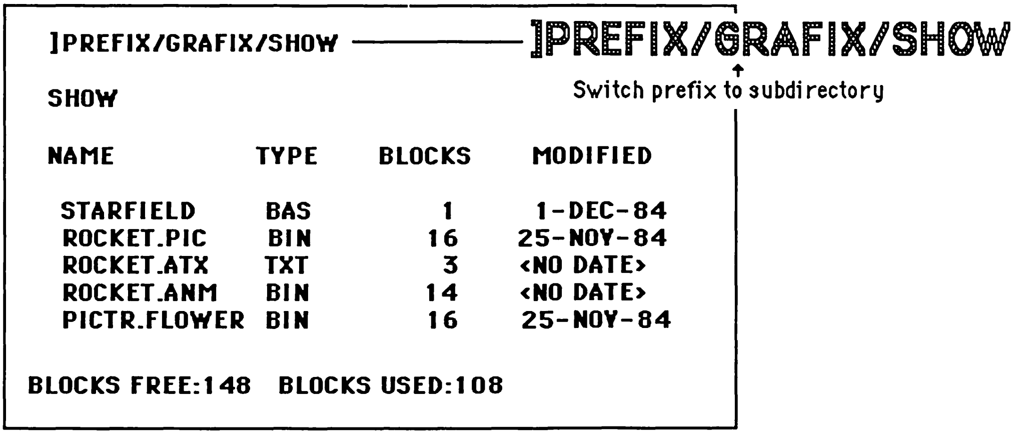
```
]PREFIX /
]CAT

/GRAFIX

NAME          TYPE    BLOCKS    MODIFIED

*PRODOS        SYS       31      1-JAN-84
*BASIC.SYSTEM  SYS       31     15-JAN-84
*STARTUP       BAS        1     25-NOV-84
 SHOW          DIR        1     25-NOV-84


BLOCKS FREE:148   BLOCKS USED:108
```



```
]PREFIX/GRAFIX/SHOW

SHOW

NAME          TYPE    BLOCKS    MODIFIED

 STARFIELD     BAS        1      1-DEC-84
 ROCKET.PIC    BIN       16     25-NOV-84
 ROCKET.ATX    TXT        3     <NO DATE>
 ROCKET.ANM    BIN       14     <NO DATE>
 PICTR.FLOWER  BIN       16     25-NOV-84

BLOCKS FREE:148   BLOCKS USED:108
```

How do I name a file?

The rules for naming files differ in DOS 3.3 and in ProDOS.

DOS 3.3

DOS 3.3 allows names that are up to 30 characters long. The name must begin with a letter but may contain numbers and all punctuation marks except commas. Only uppercase characters are accepted. Here are some examples:

MYPIC	Good
MY PICTURE	Good
MY FIRST PICTURE	Good
PICTURE.ONE	Good
PICTURE.1	Good
1ST PICTURE	No: doesn't start with letter
HOME,HOME ON THE RANGE	No: contains comma

ProDOS

In ProDOS, names can only be up to 15 characters long, must begin with a letter, and cannot contain spaces. Periods are the only punctuation marks allowed. Only the first, fourth, and fifth example in the DOS 3.3 section above are valid under ProDOS.

ProDOS filenames are sometimes called pathnames because the actual filename is only a portion of the *entire* name of the file. The full filename includes *all* directories in which that file is logged. Most of the time, this will be the name of the disk on which the file is found. MYPIC's full name then, might be /GRAFIX/MYPIC. If you've created a subdirectory, the full name might be /GRAFIX/HIRES/MYPIC. Pathnames can be up to 64 characters long.

Both DOS 3.3 and ProDOS have a special filename—the program with this filename automatically runs when the disk is booted. In DOS 3.3, it's any name used when the disk is initialized—by tradition, HELLO. In ProDOS, the startup program is always called STARTUP. Only programs, never pictures or other data, should use these names.

When naming your files, choose short names that make sense to you. ProDOS forces you to do this, but the long filenames accepted by DOS 3.3 can tempt you to use a name that's hard to remember and that can lead to errors when loading files. Find a balance between length and meaning. Note also that some graphics utilities, especially under DOS 3.3, append a suffix or prefix to the name you specify. This addition will not be seen unless you catalog a disk under DOS.

What's a data disk?

A data disk is a disk used to save information: pictures, fonts, shapes, programs, and so on. Data disks may be bootable or not bootable depending on how you initialize the disk.

DOS 3.3

In DOS 3.3, the INIT command always creates bootable disks. It does this by saving DOS and a basic program (the HELLO program) to the disk. This leaves you with approximately 120K of disk space (496 sectors) for storage of data.

Software that offers disk initialization as a menu or command choice may format the disk without saving DOS and a HELLO program to the disk. This gives you the entire 140K of disk storage for your data. Don't try to boot these data disks—they don't contain the DOS image necessary for booting.

ProDOS

ProDOS does not create bootable disks. It creates 140K data disks unless you explicitly copy the system files PRODOS and BASIC.SYSTEM to the disk to make it bootable. The system files necessary to boot the disk take up 26K of disk storage.

We prefer to work with bootable data disks as we find it more convenient when reviewing or showing our work. The extra 20 or 26K of storage on nonbootable data disks isn't valuable enough to outweigh the convenience of being able to boot a disk.

What's a slave disk?

The differentiation between master and slave disk is, at this writing, only important for DOS 3.3. The normal INIT command saves DOS as a snapshot of memory. When the slave disk is booted, the snapshot is loaded back into memory, disturbing only a small part of the total Apple memory.

A master disk is created with the Master Create program on the DOS 3.3 System Master and boots differently. When a master boots, DOS is loaded very low in memory. The boot process pushes DOS higher and higher until it hits the highest RAM address. The boot, then, destroys any information that might have been in memory. Slave disks will only boot if there is as much or more RAM as the system on which it was INITed. Master disks will boot on any size Apple. Because it is highly unlikely that your disks will be booted on an Apple with less than 48K, don't bother creating master disks.

How do I see what's on a disk?

The list of file names and other associated information on a disk is called a catalog.

DOS 3.3

You can get a disk catalog by typing CATALOG RETURN. This command lists the files on the disk that's in the drive you're currently using. To catalog a disk in the other drive, type ",D1" or ",D2" after the catalog command, before you press RETURN. If the catalog is longer than one screenful, the listing will pause so you can read it. Press any key to continue the listing. Wait for the Applesoft prompt (]) before keying in another command.

ProDOS

You can get a disk catalog by typing CAT RETURN. If you're working on an 80-column screen, typing CATALOG RETURN will show the catalog with more information about each file. To catalog a disk in another drive, you can specify the drive ",D2" or specify it's pathname. When a disk is booted, ProDOS stores its name. All ProDOS commands are routed to this disk name (unless you override it with the ",D1" and ",D2" syntax). The disk name is officially termed a pathname. (See How do I name a file? p. 60) If your boot disk is called "/BOOT" and the disk you want to catalog is "/GRAF1", typing CAT /GRAF1 RETURN will catalog /GRAF1 regardless of the drive it's in.

ProDOS allows the creation of subdirectories. These can be thought of as catalogs within catalogs. Like data, subdirectories are stored as files on the disk with a file type as DIR. On the full disk catalog, you won't see the data files held in a subdirectory, only the subdirectory's name. To specify a subdirectory, extend the pathname to include the name of the subdirectory. For example, if /GRAF1 has a subdirectory called /LORES, you can list the files in this subdirectory by typing CAT /GRAF1/LORES RETURN.

When you're using commercial software, the program usually gives you a catalog disk option. Consult the documentation of that program to find out how to use that option.

How do I remove a file from a disk?

When you accumulate disks full of programs, pictures, routines, etc., you should get into the habit of "diskkeeping". The procedure is analagous to housekeeping but instead of old newspapers and such, you discard and reorganize files.

Since you're encouraged to save your pictures during the stages of their creation, you can easily end up with a disk that contains many versions of the same picture. Once the picture has been satisfactorily completed you won't need all these versions and the 33 sectors or 17 blocks of disk space that each one is using is too valuable to waste. At this point, you can "throw out" the unneeded files.

In DOS 3.3, the command DELETE *filename* will eliminate a file from your catalog listing.

In ProDOS, the command is the same: DELETE *pathname.*

Be sure that the filename or pathname is typed exactly as it appears on the disk catalog. Don't delete a file unless you're sure you no longer want it. Consider the action irreparable although, in actuality, a deleted file can sometimes be retrieved.

When you have many files to delete, you can use FID on the DOS 3.3 system master diskette, or the Filer on the ProDOS diskette, and select the Delete Files option.

In DOS 3.3, when you delete a file, DOS marks the file as deleted in the catalog, but the file is still on the disk and will be there until you save a new file. The very next file you save to that disk will replace the file that DOS was last instructed to delete. If you haven't saved a new file to that disk, it's possible that the file you deleted can be restored. To do this, you need a special DOS utility such as BYTE ZAP on Apple Mechanic from Beagle Bros. Under ProDOS, file recovery is possible but more difficult than under DOS 3.3. Unless you know what you're doing file recovery utilities can irrevocably damage your disks!

In the world of computers, the converse of an old adage holds true: When in doubt, don't throw it out!

What's an I/O Error?

An I/O error results from attempting to read or write to a disk that DOS can't interpret. If you see the message I/O ERROR on the Apple screen your problem is one of the following:

1. The disk isn't initialized.

   Disks must be formatted and initialized by DOS and ProDOS before you can use them. Uninitialized disks generate I/O errors because they haven't been organized into Apple DOS's track and sector format.

2. You are using the wrong DOS.

   ProDOS cannot read DOS 3.3 disks and vice versa—the information is stored differently. Apple Pascal and CP/M disks are also unreadable be each other or by DOS. Try the disk with another operating system before you assume it's not initialized.

3. You lost the disk to dust, magnetic fields, fingerprints, or other demons.

   Microscopic damage to the disk, especially where the catalog information is stored, can render the entire disk unreadable even though most of the data is intact. You may be able to recover some of the information by using a utility program like Bag of Tricks from Quality Software, but chances are you won't be able to recover all of the data. You did remember to keep backup copies of your software and data, didn't you?

For a list of other DOS, ProDOS, and Applesoft errors see the Appendix p. 283.

What's a fast DOS?

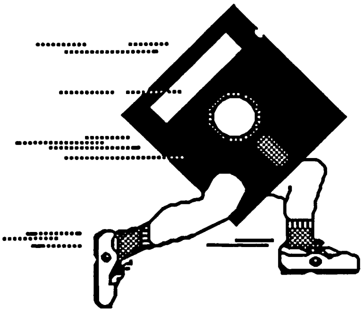
When we first start working with computers we're impressed with the speed with which they do things. After a while however, we become spoiled and no matter how fast something is done we want it done faster. In the world of computers, seconds seem like hours. To the artist, nothing seems to take as long as it takes to BLOAD binary files—pictures, shape tables,and so on. Enter fast DOS.

DOS 3.3 is not efficient when instructed to BLOAD a file. Rather than loading the file directly to the appropriate memory address, it first loads to a buffer, an unused area of memory reserved for just this purpose. When the buffer is full, the data is moved to the final location. Many of these loads and moves are required to fully load a picture.

By rewriting parts of DOS, binary files can be loaded directly to the target addresses. Rewriting DOS is a task for accomplished assembly language programmers. Fortunately, DOS speed-ups are available commercially.

Commercial fast-DOS packages include additional features and utilities such as the option to create nonbootable data disks, free-sector counts, and RAM disks. (See What's a RAM disk? p. 66) Most of these packages include routines to convert disks already initialized under standard DOS 3.3 to fast DOS. You may not be able to do this with commercial graphics software, especially copy-protected software, but you can use the fast DOS on your graphics data disk. With a fast DOS on your data disks, you'll have much more control over how a graphic presentation is paced.

ProDOS already loads binary files at top speed and, at this writing, there are no ProDOS enhancement utilities.



What's a RAM disk?

A RAM disk is a simulated disk created out of the computer's RAM. A RAM disk works exactly like a normal floppy, but because the disk is really RAM, the simulated disk is fast—blazingly fast.

RAM, however, is forgetful. The information stored in RAM dissipates when the power is turned off. When working with a RAM disk then, you must remember to save all files to a standard floppy before turning off the computer. If the electricity goes out before you save, your work will be lost.

DOS 3.3

On all Apple computers, with the proper software, the language card area can be used as a small RAM disk.

For the Apple II, IIplus, and //e computers, interface cards containing extra RAM may be purchased to create RAM disks as large as 320K, almost the size of two and a half standard Apple floppies! These interface cards may also be used for data storage or even program space by some specialized programs.

On a standard //c and Apple //e with the Apple extended 80-Column card (or equivalent), you can create a RAM disk with approximately 60K of data space by using a commercial RAM disk utility.

ProDOS

ProDOS users have it easier. The current release of ProDOS, version 1.1, automatically creates a RAM disk on Apple //c and 128K //e computers. To access this drive, use the prefix "/RAM". The ProDOS command PREFIX /RAM will set the standard prefix to /RAM and all further ProDOS commands will be directed to the RAM disk until another prefix is set.

Regardless of the DOS, be aware that both Aux RAM disks and double-high-resolution graphics use the same area of memory. They're not easily used together.

Graphic Bonus: RAM Disk Slide Show

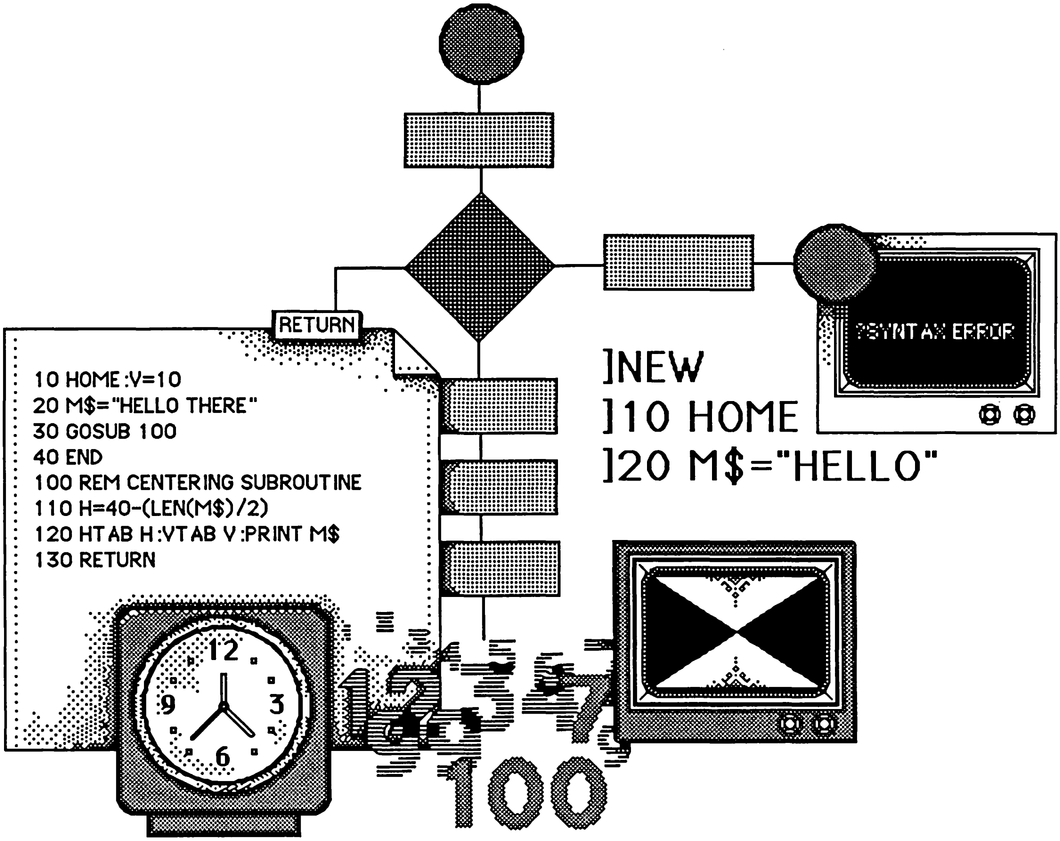
This program will only work on 128K Apple //e and Apple //c computers running a version of ProDOS that automatically creates a RAM disk. Because RAM disks have only 120 free blocks, you can copy only seven files to the RAM disk. Be sure you're in 40-column active mode—this program uses page flipping which doesn't work with the 80-column display.

```
10 TEXT:HOME
12 VTAB 23:HTAB11:PRINT "Loading to RAM disk"
15 GOSUB 200
20 L1=8192:L2=16384:LOC=L1
30 READ F$
40 IF F$="DONE" THEN GOTO 500
50 PRINT CHR$(4);"BLOAD";F$;",A";LOC
60 C=C+1:IF C=1 THEN POKE -16297,0:POKE -16302,0:
   POKE -16304,0
70 IF LOC=8192 THEN POKE -16300,0:LOC=L2:GOTO 90
80 POKE -16299,0:LOC=L1
90 GOTO 30
200 PRINT CHR$(4);"PREFIX /RAM"
210 READ P$
220 READ F$
230 IF F$="DONE" THEN RESTORE:READ P$:RETURN
240 PRINT CHR$(4);"BLOAD ";P$;"/";F$
250 PRINT CHR$(4);"BSAVE /RAM/";F$;",A$2000,L$2000"
260 GOTO 220
500 PRINT CHR$(4);"PREFIX ";P$
510 END
1000 DATA /prefixdisk1
1010 DATA mypic1.pic
1020 DATA mypic1.pic
1900 DATA DONE
```

Change line 1000 to contain the volume name of your picture disk and lines 1010 and following to contain the names of your pictures. The last "name" in the data statement must be DONE—this is what signals the end of the slide show. Remember you can store only seven pictures in the RAM disk.

You'll probably need a delay loop at line 85—it's fast.

If you're animating, you may want the program to replay. If so, change line 500 to read RESTORE:READ P$ and change line 510 to read GOTO 30. With these alterations, the program will repeat until you press CONTROL C or CONTROL RESET to stop the program.



```
10 HOME:V=10
20 M$="HELLO THERE"
30 GOSUB 100
40 END
100 REM CENTERING SUBROUTINE
110 H=40-(LEN(M$)/2)
120 HTAB H:VTAB V:PRINT M$
130 RETURN
```

```
]NEW
]10 HOME
]20 M$="HELLO"
```

About Programming

About programming

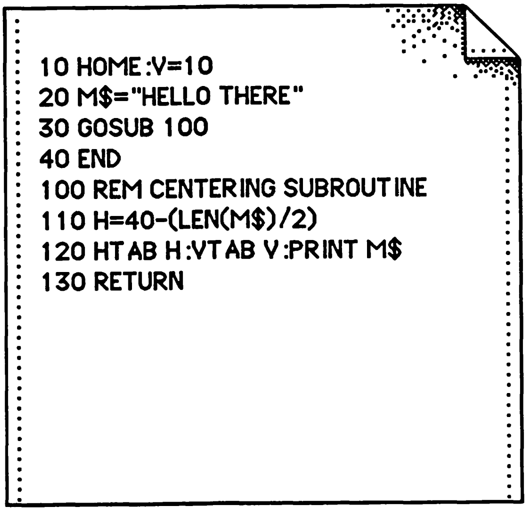
What's a computer program?

A program is a list of instructions for a computer. The catch is that the instructions must be written in a language the computer understands.

Computer languages are restricted languages. Applesoft BASIC contains a vocabulary of slightly over 100 words or instructions. Like human languages, computer languages have a grammar and syntax. Unlike human languages, computer grammar and syntax is very simple. Statements cannot be ambiguous and must be very carefully sequenced in relation to the other statements in the program. Learning to program means learning the available instructions and the syntax associated with a language that will accomplish a task.

Programming is a problem-solving activity. Programs are written to accomplish a goal: show a sequence of pictures, animate a figure across the screen, make a drawing based on joystick input, and so on. For a program to be written, the problem must first be divided into small steps, each of which accomplishes a small part of the overall goal. The programmer is concerned with sequencing these steps correctly and making sure that each piece communicates correctly with other pieces.

It can be as much fun to program as it is to use other's programs!



```
10 HOME:V=10
20 M$="HELLO THERE"
30 GOSUB 100
40 END
100 REM CENTERING SUBROUTINE
110 H=40-(LEN(M$)/2)
120 HTAB H:VTAB V:PRINT M$
130 RETURN
```

What's a programming language?

The CPU is operating on wires that are either carrying current or aren't. The CPU, then, must receive it's instructions in binary code.

The earliest computers were programmed only in binary or machine language. Most people have difficulty working with binary. It's easy to mistakenly type 01101001 when you mean 01001001.

Assembly language was created to solve some of the problems with programming in binary. Assembly language substitutes mneumonics for each of the binary instructions—ADC rather then 01101001. The assembly language program is then submitted to an assembler which translates the mneumonics into the binary code that the CPU understands.

Assembly language is still popular; it produces the fastest running programs. However, because there is a one-to-one correspondence between the mneumonics and the binary, assembly language is tedious. High level languages were created to remove some of the tedium and make programming available to people unwilling to learn assembler.

High-level languages are more like English. Instead of mneumonics, they feature full words—PRINT, INPUT, WRITE, and so on. Like assembly language, high-level languages must be converted to machine language before the CPU can execute the instructions.

There are two ways to accomplish this translation. One is to wait until the entire program has been written and do the translation in one fell swoop. Languages that operate in this manner are termed compiled languages. The alternative is to have the computer translate each line of code as the programmer types it. Languages that operate this way are called interpreted languages. The translators are programs themselves, programs whose sole function is to translate other programs into machine code.

Nearly every existing computer language—both interpreted and compiled—has been adapted to the Apple.

What's BASIC?

BASIC is a programming language developed in the mid-1960s at Dartmouth College. At that time, most computers operated in batch mode; programs were submitted to the computer in batches on punch cards. The programmer did not interact with the computer on a CRT.

At Dartmouth, an interactive computer system was developed and BASIC was the primary programming language used on the system. When personal computers were invented, some ten years later, BASIC was the most common interactive language and was quickly adapted for personal computers.

Steve Wozniak, one of the founders of Apple Computer, wrote a verson of BASIC called Integer BASIC for the first Apple computer. Integer BASIC was a challenge. Steve wanted to prove that the then popular arcade game, "Breakout," could be created on an Apple in BASIC. Integer BASIC is fast but has many limitations—it can deal only with whole numbers and it has no direct access to the Apple's hi-res graphics. To provide a more powerful BASIC, Apple contracted MicroSoft to adapt their popular BASIC to the Apple. Applesoft was born.

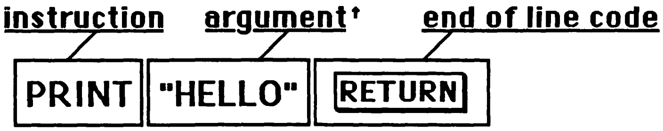
Applesoft, with decimals and hi-res graphics, quickly became the "standard" Apple BASIC.

All Apple owners can still run Integer BASIC by loading it into the Language Card of the Apple IIplus or bank-switched memory of the Apple //e or //c. The DOS 3.3 System Master Disk does this automatically when it's booted. Many interesting lo-res graphics demonstration programs are written in Integer BASIC, but most programs have been translated or adapted to Applesoft. At this writing, Integer BASIC cannot be used with ProDOS.

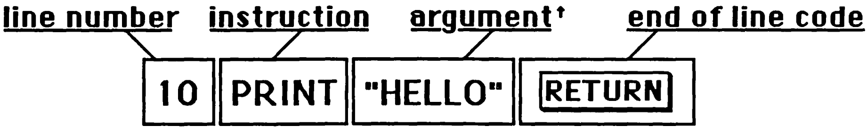
Most BASICs look pretty much the same, differing primarily in the number and capabilities of the instructions included in the language. Anatomy of BASIC (p. 74) will show you the appearance of a line of BASIC code and the overall structure of a BASIC program.

Anatomy of BASIC

Immediate mode (typed directly at the prompt)



Deferred mode (with line number - in a program)



† may be optional depending on instruction

10 HOME 20 V = 10 : CT = 40 : D = 50 30 M$ = "GRAPHICS BY ARTHUR DENT"	initialization
40 NORMAL : GOSUB 100: GOSUB 200 50 INVERSE : GOSUB 100 : GOSUB 200 60 C = C+1 : IF C = 50 THEN GOTO 500 70 GOTO 40	main program
100 H = (CT - LEN(M$))/2 110 VTAB V : HTAB H : PRINT M$ 120 RETURN 200 FOR T = 1 TO D : NEXT T 210 RETURN	subroutines
500 NORMAL 510 END	program end
1000 REM Program to simulate FLASH 1010 REM in eighty-column-active mode 1020 REM for Apple //e and //c computers.	documentation

Will the same programs work on all Apples?

For the most part yes, although each Apple is unique. Programs that capitalize on a special characteristic of a particular machine will not work on another.

There is no way to tell if an older program will work on the latest Apple without trying it, but newer commercial programs generally specify their requirements. Software that use double hi-res graphics cannot run on an Apple II or Apple IIplus—these packages are labeled "Apple //c" or "128K Apple //e."

Programs that used special coding to compensate for the simpler keyboard of the Apple II and IIplus computers may not work correctly on the Apple //e or //c computers. Fortunately, most graphics programs (except for double hi-res) can be used on all Apple II computers and the best of the older programs have been revised to operate on the current Apples.

Apple II programs will not run on the Apple 32 line of computers. The Lisa and Macintosh computers have different microprocessors and operating systems.

Most DOS 3.3 Applesoft programs for the Apple II and IIplus will run on "Apple compatible" computers such as the Franklin or Basis. ProDOS must be modified to run on these computers and programs which depend on special features (such as double hi-res) of Apple //e and //c computers will not run correctly.

Programs for other computers—IBM PC, Atari, Commodore—will not run on the Apple. In fact, disks created by any one of these computers cannot be read by another. The Atari 800 and Commodore 64 computers are similar to the Apple and many companies have created versions of their programs for all three computers.

Graphics rely heavily on machine-specific details—how the hi-res screen is constructed, how colors are coded, and so on. Therefore, it's difficult, if not impossible, to transfer most graphics from computer to computer. Add-on devices for some computers may enable you to run Apple software on them. These peripherals, currently available for the Commodore 64 and the IBM PC, transform the host computer into an Apple. A similar device can make the Apple act like an IBM PC.

Do I have to learn to program?

We couldn't decide how to answer this one. Flip a coin and pick your answer.

No. There's a vast selection of commercial Apple programs to do nearly anything you would want to do.

Yes. Commercial software can't give you complete freedom to create exactly the image or animation you desire. Learn to program and you can get more out of the computer.

Are you still waiting for an answer?

Actually, we believe you should become comfortable enough with BASIC programming to write and use small programs. Most graphics software authors include simple programs that you can use to display the graphics created with their package. For the most part, all you have to do is type these programs in and you're off and running. However, if you understand a little about what these programs are doing, you'll have greater control of your graphic presentations.

If programming doesn't interest you, feel free to leave the "heavy stuff' to the professionals.

How do I learn to program?

The only way to learn to program is to write programs.

We found it very helpful to enter programs from magazines and books, modifying them to suit our particular goals. This enabled us to see how working programs were constructed and to get a sense of how to structure our own programs.

At some point, you may feel the need to take a formal course in programming. Many universities offer adult education courses in computer programming. Informal courses in programming are often offered by local Apple user's groups.

The programs in this book will provide you with a good start. If you want to learn more about programming, check out the bibliography in the Appendix (p. 283).

How do I enter a program?

1. Place a DOS 3.3 or bootable ProDOS disk in disk drive 1 and turn the computer on. Wait for the Applesoft prompt(]).

2. Type NEW and press RETURN .

   NEW clears out any program currently in the computer's memory. Pressing the RETURN key tells the computer that you have finished entering your command.

3. Type the first program line.

   In Applesoft BASIC, all program lines must be numbered. While it's customary to enter lines in numerical order, it's not essential. (BASIC will automatically rearrange them in order.) Programmers usually number their program lines in increments of 10 (10, 20, 30...) so that new lines can be inserted at any time without retyping the entire program.

   If you misspell a BASIC command, or omit a required punctuation mark, BASIC will not tell you that there's an error (SYNTAX ERROR) until you try to RUN the program. Therefore, after you type a line, be sure to proofread it. Typing errors can be corrected by using the ← and typing over the error. After you've made the correction, use the forward arrow key → to trace over the rest of the line.

4. When you're sure the line is entered correctly, and the cursor is positioned at the end of the line, press RETURN . RETURN will enter the line into your program. Until you do this, the line is stored in a temporary location or buffer.

5. Repeat instructions 3 and 4 for each program line.

6. Type LIST.
   LIST will instruct BASIC to show you the program currently in memory (in this case, the one you just typed.) Once again, proofread the typing. If you find an error, retype the incorrect line using the **same** line number. BASIC will replace the incorrect line, with the newly typed one. If a program is longer than one screenful,(24 lines), you can LIST a few lines at a time. The command, LIST 50,85 will list only lines 50 through 85. The command, LIST ,30 will list all program lines up to line 30. The command, LIST 30, will list program lines from 30 on.

How do I execute or see a program?

After you've entered and proofread a program, type the command RUN, and press RETURN.

BASIC will start at the lowest line number and execute each instruction as it comes to it. When it runs out of lines, or reaches an END instruction, it will stop. If there's an error in the program, BASIC will report it to you when it tries to execute the erroneous instruction.

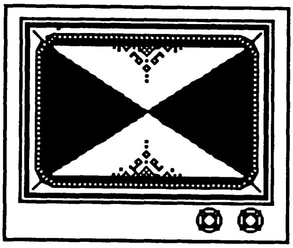
The most common error is SYNTAX ERROR IN LINE nn, where nn is the line number that contains the error. When this happens, LIST the offending line number with the command LIST nn, and retype it correctly. (See How can I edit a program line? p. 79)

As you enter more complicated programs, you'll probably get other error messages; to correct these, you'll need some knowledge of BASIC.

To stop a program in midstream, press CONTROL C. If the program is waiting for you to enter information, follow the CONTROL C with RETURN.

Here's a short program for you to enter and run:

```
10 HGR: POKE -16302,0
20 HCOLOR=INT(RND(1)*7)+1
30 FOR I = 1 TO 20
40 FOR X = 0 TO 279 STEP 2
50 HPLOT X,0 TO 279-X,160
60 NEXT X
70 NEXT I
```



Why am I getting error messages?

If you got the BASIC error message ?SYNTAX ERROR, you've probably misspelled a command, omitted a required colon, or something like that.

If your error message doesn't have a question mark in front of it, for example NOT DIRECT COMMAND, it's a DOS error. The most common DOS error is FILE NOT FOUND (or PATH NOT FOUND in ProDOS). You've probably misspelled a file or path name.

Integer BASIC errors have three asterisks preceding the error message, for example ***SYNTAX ERROR. These errors are similar to Applesoft errors.

Errors have numbers assigned to them. If, when using a program, you get an error message such as: ERROR #8, you'll have to look up the error. ERROR #8 is an I/O ERROR.

You'll find a list of all errors in the Appendix (p. 283).

How can I edit a program line?

When BASIC indicates a programming error it tells you what the error is and what line it occurs in, e.g. SYNTAX ERROR LINE 30. LIST the line to find the error.

Needless to say, you must identify the error before you can correct it. The simplest mistakes to find are typing errors. Most of your errors will probably be of this type. Check the spelling of the commands and the syntax. You may have more trouble identifying logical errors.

Once you know what's wrong, you can simply retype the line. BASIC will replace the old line with the new, error-free line. To delete a line altogether just type the line number and press RETURN. To delete a range of lines type DEL n1,n2 where n1 is the first line you want to eliminate and n2 is the last line. These lines and everything inbetween will be deleted.

Once your program is corrected, don't forget to SAVE it to disk. (See How do I put a program on disk? p. 80)

After you've had some experience with the Apple, you may want to learn how to do screen editing. There are ways to move the cursor around the screen to correct errors. Consult the Applesoft Programmer's Manual for more information.

How do I put a program on disk?

As you enter—type in—a program, it's stored in Apple's RAM. To save it permanently, you must put it onto a disk.

To save to disk, you must boot DOS or ProDOS before entering the program and you must have an initialized disk for the correct DOS at hand. Once the program is in memory and your initialized disk is in the drive, type SAVE name RETURN. The program name must follow the file name conventions of the DOS you're running. (See How do I name a file? p. 60) The disk will whirr in the drive as the program is saved to it. Later, if you make changes to this program, they'll be made only to the RAM version, not the disk version, so be sure to save it again.

If you save a program using the same name as a program already on the disk, the new version replaces the old version on the disk. Neither DOS nor ProDOS warn you that a file name has been previously used. When you have many programs on a disk, catalog the disk before saving any new programs to it. In this way you'll avoid using a file name you've used before.

How do I get a program from disk?

You can get a program from disk by typing LOAD name RETURN, where name is a program name on the disk's catalog.

The program you load must be a BASIC program. In DOS 3.3, a BASIC program will have an A or an I in the file type column of the catalog; in ProDOS there will be a BAS in the file type column. You can't LOAD a T (text) or B (binary) file. To load an I file, you must have Integer BASIC in your computer.

The LOAD command brings the program from the disk into RAM; it doesn't execute it. Once loaded you can LIST and modify the program or you can RUN it.

A program can be retrieved and executed in one step by adding the file name to the RUN command. RUN name RETURN tells DOS to load the program and immediately run it. It's equivalent to typing LOAD name RETURN followed by RUN RETURN.

Be careful. When the program you want to execute is already in memory, use RUN RETURN, *without* a filename. Work can be lost by accidentally using RUN name RETURN.

What's a variable?

Programs must respond to changing conditions. In a drawing program, for example, a joystick may provide constantly changing X and Y coordinates. A program should allow you to save your picture under a name of your choosing rather than a predefined name. To allow this flexibility, BASIC allows the programmer to assign and retrieve values to and from RAM memory *by name*. These names are called variables.

How do I use variables?

One way to define a variable is by using the BASIC instruction LET. LET A=10, for example, assigns the number 10 to the variable A. After this assignment, the value stored can be retrieved by referring to the name of the variable. PRINT A, in this case, will print 10 on the screen.

You can change the value of any variable whenever the program calls for a change. Try the following program. (Type your name surrounded by quotation marks where indicated in line 10.)

```
10 LET NA$="your name"
20 PRINT "HELLO ";NA$;" NICE TO MEET YOU."
```

With this program you must retype line 10 to change the name. Another way to get information stored in a variable is to use the INPUT instruction. Change the program to read:

```
10 INPUT "TYPE YOUR NAME";NA$
20 PRINT "HELLO ";NA$;" NICE TO MEET YOU."
```

When you run this program, the computer will print "TYPE YOUR NAME" and then wait for you to enter information. When you respond by typing in your name and pressing RETURN, whatever you typed will be stored under the variable NA$. In line 20, BASIC will retrieve the stored name and PRINT it.

INPUT and LET are the primary methods of assigning a value to a variable. (See What's READ/DATA? for another way. p. 84)

By the way, you don't have to type LET. When Applesoft BASIC sees a variable name on the left side of an equal sign, it assumes you are making an assignment. Therefore, LET A=10 and A=10 are equivalent statements.

How do I name a variable?

Variable names must start with a letter. They can contain letters or numbers but not punctuation marks. In Applesoft they can be as long as you want them to be, but only the first two characters are used—the others are ignored. Furthermore, variables may not contain any Applesoft commands. Here are some examples of acceptable and unacceptable variable names:

A	good
A1	good
NAME	good—same as NA
A99	good—same as A9
9A	no good—starts with number
GRID	no good—GR is an Applesoft command
COLOR	no good—COLOR and OR are Applesoft commands
MY NAME	no good—contains a space
MY.NAME	no good—contains punctuation

Because programs can get complicated, it's easy to lose track of your variables. Choose a variable name that reflects its function. Assign the length of a delay to the variable D, store color in C, the number of pictures in NP, and so on.

Besides using a legal name when naming variables in BASIC, you must use a variable name that indicates what kind of information you're storing in that variable. (See Are there different kinds of variables? p. 83)

Are there different kinds of variables?

Programs store different types of information: names of files, numbers, and so on. BASIC must know the kind of information you intend to store in a variable. BASIC recognizes three simple variable types: integers (whole numbers), reals (decimal numbers), and strings (words or letters). The data type is designated by the variable name.

Integers have a percentage sign: A% is an integer variable.
Reals have no symbols: A is a real variable.
Strings have dollar signs: A$ is a string variable.

If you assign the wrong type of data to a variable, the program won't work right and you might get a syntax error. Here are examples of acceptable and unacceptable assignments:

Statement	**Approximate value stored**[1]
A=10	10.000000
A=2.5	2.5000000
A%=2	2
A$="PIC"	"PIC" (note the quotation marks around strings)
A%=2.5	2
A=TEN	the current value of TE—the N is ignored
A="TEN"	nothing—SYNTAX ERROR
A$=12	nothing—SYNTAX ERROR
A$= MYPIC	nothing—SYNTAX ERROR

You can use variables and assignment statements to do mathematics:

A=2+3	5.0000000
A=2-3	-1.000000
A=2*3	6.0000000 (* indicates multiplication)
A=6/2	3.0000000 (/ indicates division)
A=2^3	9.0000000 (^ indicates "to the power")

What's an algorithm?

An algorithm is a formula. It can be a simple, one-line formula to center text on the screen or a several-page program to do a special sort on a file. Algorithms are often used to generate the value of a variable. $C^2 = A^2 + B^2$ is an algorithm you may be familiar with.

[1] With mathematical operations on real numbers, computers can make rounding errors.

What's READ/DATA?

When there's a lot of data in a program, LET and INPUT are inadequate methods of assigning a value to a variable. There are several other methods to store and retrieve large quantities of data. The companion instructions READ/DATA are probably the most useful of these methods.

Each time BASIC finds a READ statement, it looks for an item in a DATA statement. BASIC keeps track of where it is in the DATA list by usng a special data structure called a pointer. A pointer is a datum that stores the location of another piece of data. Each time a new READ instruction is executed, the pointer is incremented to indicate the *next* item on the data list. If there's no more data, BASIC reports an OUT OF DATA error. (A conditional statement can be added to avoid an OUT OF DATA error. IF A$=... THEN ... See What's a branch? p. 89.)

Examine the following program, which prints a list of soups:

```
10 FOR I = 1 TO 4
20 READ A$
30 PRINT A$
40 NEXT I
100 DATA VEGETABLE,TOMATO,CHICKEN
110 DATA CLAM CHOWDER
```

Compare it to the equivalent program which uses LET:

```
10 LET A$=" VEGETABLE"
20 PRINT A$
30 LET A$=" TOMATO"
```

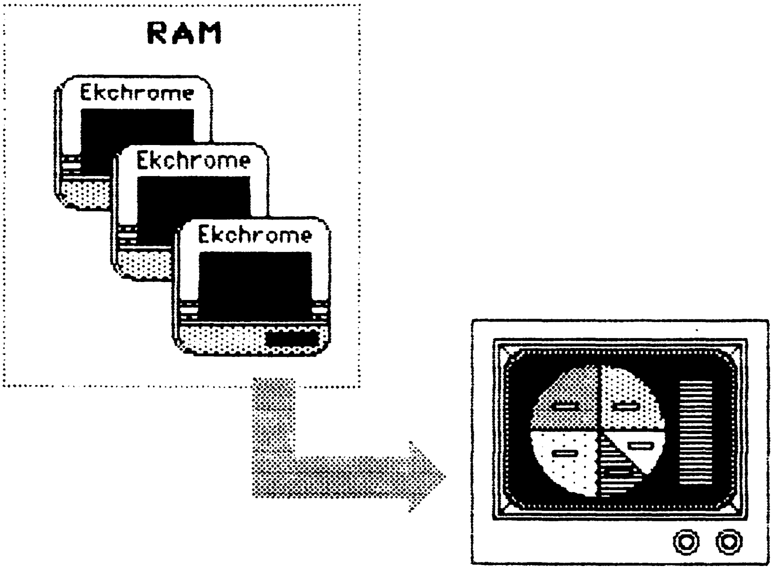
```
40 PRINT A$
50 LET A$=" CHICKEN"
60 PRINT A$
70 LET A$=" CLAM CHOWDER"
80 PRINT A$
```

The program using READ/DATA is simpler and more flexible. To change from soup to nuts, for example, all you'd have to do is edit two lines.

The instruction RESTORE resets the pointer to the beginning of the data list. Add this line to the first program:

```
35 RESTORE
```

Only VEGETABLE will be printed because, upon executing RESTORE after each READ, BASIC resets the pointer.



What's a counter?

A counter is a type of variable used for enumeration.

When you see LET A = A + 1, you're seeing a counter in action. This statement, better than any other, illustrates the fact that computer programming is not mathematics—in math, A cannot equal itself plus one.

How does BASIC do it? It retrieves the value of A, adds one to it, and then stores the new value back in A.

Counters are very common in graphics programs because we frequently must know how often something has happened.

Try the following program to see how counters work:

```
10 LET A = 1
20 PRINT A
30 LET A = A + 1
40 IF A > 10 THEN END
50 GOTO 20
```

Imagine that instead of PRINT A, line 30 was a statement which caused an animated man to take one step. The program would make the man walk 10 steps.

Bug alert. One of the most common programming errors is forgetting to initialize a counter. BASIC will start the counter from 0. In the above program, line 10 initializes the counter. If you want the count to be accurate, be sure that the starting value of the counter variable is correct.

What's a flag?

Another common special variable is a flag. A flag is a variable used by a programmer to determine if something has happened. Usually, flags can have only two values, often, but not always, 1 and 0. A paint program, for example, might use a flag to indicate whether the brush is down or not. In combination with IF/THEN branching, flags can be very powerful.

Most programs for beginners don't use flags, however in the GRAPHIC BONUS:SLIDE SHOW (p. 166) program we used a flag (called LOC) to let the program load pictures on alternate hi-res pages while displaying the other page.

What's a loop?

A loop is a portion of a computer program that's repeated a specified number of times.

One way to loop in BASIC is to use a GOTO instruction. Consider the small program from What's a counter? (p. 86) The GOTO statement in line 50 of that program caused BASIC to jump back to line 20 and re-execute all program statements inbetween another time. Note also that without line 40, the program would never stop. (See What's a Branch? p. 89 for more information about line 40.)

Another way to loop is to use the FOR/NEXT statement. You can think of FOR and NEXT as brackets—anything between them will be repeated. Here's an example:

```
10 GR : COLOR=13
20 FOR I = 0 TO 39
30 PLOT X,Y
40 Y = Y + 1 : X = X + 1
50 NEXT I
```

The FOR in line 20 and the NEXT in line 50 delimit the statements that are to be repeated—in this particular program, a PLOT statement (PLOT X,Y) and two assignment statements (Y=Y+1: X=X+1). The number of repeats is determined by the FOR statement, which is essentially a disguised counter.

Using the variable I, this statement will count from 0 to 39, but the counter will be incremented only when the NEXT statement is reached. Therefore, everything between the FOR and NEXT will be repeated 40 times. The variable name is chosen arbitrarily; any legal numeric variable name will do.

Line 20 directs the computer to count—the default being in steps of one. STEP n can be added to the FOR statement to tell the computer to count in increments of n; n can be any number. Change line 20 to the following and RUN the program again:

```
20 FOR I=0 TO 39 STEP 2
```

There is a bug in BASIC which causes a FOR/NEXT loop to be executed at least once *regardless* of the numbers in the FOR statement.

Bug Alert. If you change the value of the loop variable (I in the example above) **inside** the loop, the loop will not work correctly. When the variable is changed, BASIC will lose count and the loop may stop prematurely or continue forever.

What's a delay loop?

A delay loop is an empty loop. The computer just counts.

Delay loops are useful in many programs. For example, if you're showing a credit screen with lots of text, the viewer must have time to read the text. A delay loop will allow the time. Here's a typical delay loop:

```
100 FOR D = 1 TO 1000: NEXT D
```

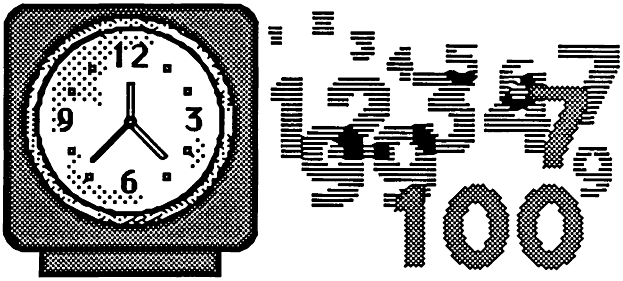
The use of the variable D for the loop is arbitrary—any legal numeric variable can be used. Upon seeing this instruction, BASIC will wait before going on to the next program statement. The larger the number, the longer the delay.

Unfortunately, we can't tell you how long this loop would wait. It depends on how long the program is and where it is in the the program. You'll find that a lot of experimentation will be necessary to get the correct delay.

Rather than retype this long line all the time, why not use a variable?

```
10 T = 1000
1000 FOR D = 1 TO T: NEXT D
```

Now you only have to retype line 10 to change the delay.



What's a branch?

BASIC normally executes programs one line after another, in ascending order. Branch statements change the execution sequence of a program.

There are two types of branches: unconditional and conditional. In a BASIC program, the unconditional branch is accomplished by the command GOTO n, where n represents a line number in the program. When BASIC executes the GOTO, it skips all program lines between the current one and the line specified by the GOTO. Too many unconditional branches can make your program hard to read. Programmers call such programs "spaghetti code."

A conditional branch allows you to have two or more courses of action based on some criterion. We do this all the time: if it's raining take an umbrella, otherwise don't. In BASIC, the most common conditional statement is IF/THEN. The syntax of this statement is IF (*condition*) THEN (*action*). *Condition* is a legal BASIC instruction that is either true or false and *action* is any legal BASIC instruction. Here's an example:

```
10 INPUT "HELLO, HOW ARE YOU?";A$
20 IF A$="GOOD" THEN PRINT "GLAD TO HEAR IT.": GOTO 40
30 PRINT "SORRY."
40 END
```

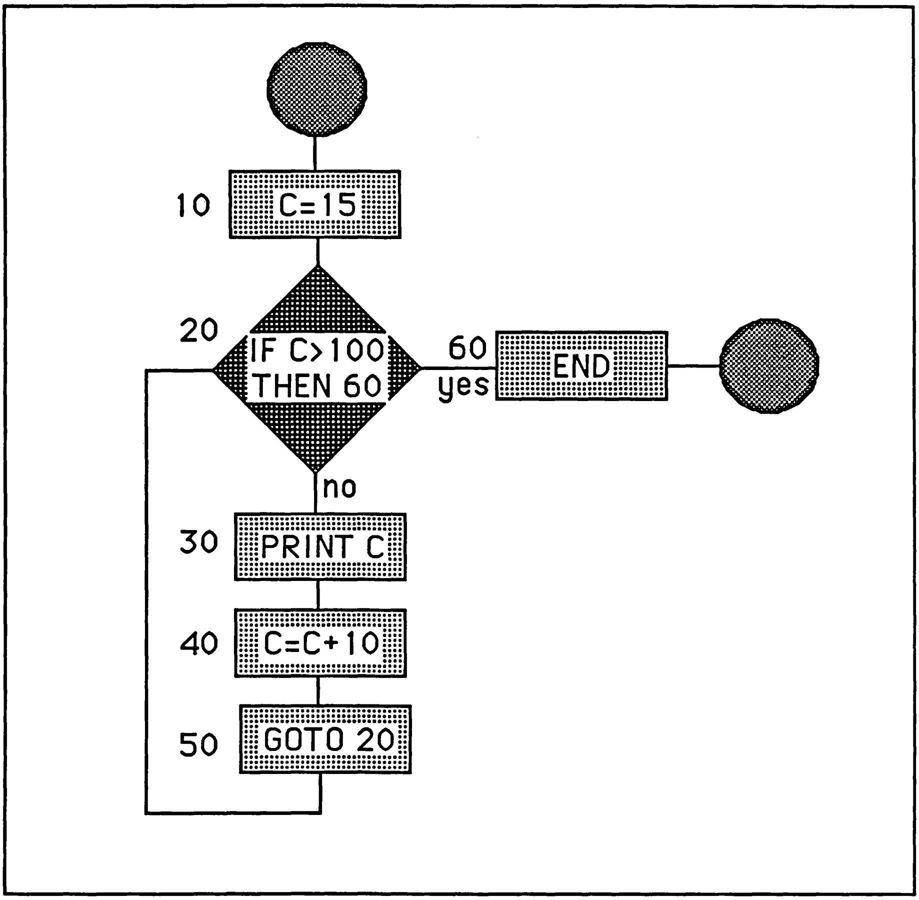
Line 10 of this program asks a question and waits for the user to type in an answer. If the condition in line 20 isn't met —the typed response is not "GOOD"—BASIC will ignore the rest of line 20 and go on to line 30—the program will offer condolences. The GOTO in line 40 is the branch. If the condition is met—the response is "GOOD"—BASIC will execute the rest of line 20, jumping to line 40 and skipping line 30. Try the program without the GOTO by retyping line 20 to read

```
20 IF A$= "GOOD" THEN PRINT "GLAD TO HEAR IT."
```

Here's a programming trick! It's often useful to know if a number is even or odd. With IF/THEN, it's easy. Try this:

```
10 INPUT "TYPE A NUMBER"; N
20 IF N/2 = INT(N/2) PRINT "EVEN": GOTO 40
30 PRINT "ODD"
40 END
```

Bug Alert. Perplexing errors can happen if you branch recklessly. Never branch **into** or **out of** a loop or a subroutine unless you know what you're doing.



What's a subroutine?

A subroutine is a subprogram. Many programs require repetitive code, code that is almost identical, in different parts of the program. Instead of having us type the same statements over and over again, BASIC allows us to extract the repetitive code and make it a subroutine.

BASIC uses the keyword GOSUB to signal a jump to a subroutine and RETURN to indicate that program execution should return to the place where it jumped from. While subroutines can make a program shorter and more understandable, they can also make it complex and confusing—it's up to the programmer to accomplish the former rather than the latter.

Consider these programs written to show a series of pictures:

1.

```
10 HGR : POKE-16302,0
20 F$="mypicl" : GOSUB 1000
30 F$="mypic2" : GOSUB 1000
999 END
1000 PRINT CHR$(4);"BLOAD"F$",A8192"
1010 RETURN
```

2.

```
10 HGR : POKE-16302,0
20 PRINT CHR$(4);"BLOAD mypicl,A8192"
30 PRINT CHR$(4);"BLOAD mypic2,A8192"
999 END
```

As written, there's little advantage to the subroutine in program 1; program 2 is just as efficient. But if you were showing 10 or 12 pictures the advantage is obvious. In program 2, line 20 would have to be repeated for each picture, resulting in a much longer program and a greater chance for errors.

Bug Alert. A common programming error is to forget to put an END statement before your subroutines. Without the END statement in program 1, the program would run normally, but then from line 30 it would go to line 1000 and load F$ (the second picture) again. Next, the program would go to line 1010 and bomb with the message "RETURN WITHOUT GOSUB ERROR."

What's a REM?

REM is an abbreviation for remark. It's an instruction which allows you to include comments, remarks, and other nonexecutable lines in a BASIC program. BASIC ignores program lines starting with REM.

Use REMs liberally to document programs. As you're writing a program you know what every line of the program code is for, but after some time passes,you may find that you've forgotten what a particular variable stands for or how a particular part of the program works; REMs take the burden off your memory.

Unfortunately, REMs also make a program longer and, if they are lengthy remarks, they may slow it's execution. If your program is too long or too slow due to REMs, you can write another version of the program leaving the REMs out. This does not mean retyping the entire program. There are several utility programs available which will strip a program of REM statements. Whenever you do this, you should save both versions of the program—one with REMs for reference, and one without REMs for execution.

Bug Alert. If you're typing in a program from a book or other publication and it contains a lot of REMs, you might be tempted to leave the REMs out. Be careful! Some programmers reference lines that contain only REMs. This means that there might be a command in the program such as GOTO 20, and line 20 may be a REM statement that you eliminated. You'll get an error message and the program won't work. Before eliminating a line that contains only a REM, read the entire program to be sure it isn't a referenced line.

How do I use DOS in a program?

While there are many differences between DOS 3.3 and ProDOS, the Applesoft programmer sees very few of them. Most Applesoft programs will run under either DOS, unless they use one of the DOS 3.3 commands that is not present in ProDOS or vice versa.

Within programs, DOS commands are written as PRINT statements. To distinguish a DOS command from a BASIC PRINT statement, a CONTROL D must precede the DOS command. Typing CONTROL D will enter a control-D character, but control characters are not displayed on the screen. So instead, we use CHR$ (4). CHR$ refers to character string. Control-D is the fourth letter of the alphabet and CHR$(n) is the BASIC instruction to print character number n.

Most programs that use DOS assign CONTROL D to a variable with the statement: D$=CHR$(4). D$ can then be used in a PRINT statement:

```
10 D$= CHRS(4)
20 PRINT D$;"BLOAD MYPIC,A8192"
```

All DOS 3.3 or ProDOS commands can be implemented in the same manner.

Special note must be made of the instructions PR#n and IN#n. These cause the Apple to send its output to slot n or look to slot n for input. These instructions are valid in both DOS and Applesoft. When used in programs, they should always be issued as DOS commands. (See How do I use a printer in a program? p. 96)

Bug Alert. Some Applesoft programs, under DOS 3.3, print a carriage return prior to the CONTROL D by defining D$ (or another variable) as CHR$(13) + CHR$(4). These programs will work only under DOS 3.3. To make them work under ProDOS, eliminate CHR$(13) by changing the statement to D$ = CHR$(4).

What are PEEK, POKE, and CALL?

The Apple's extensive system programming is not totally accessible from BASIC. Only a limited number of commands can fit into the BASIC ROM. PEEK, POKE, and CALL are BASIC instructions that give you wider access to the Apple system.

All three commands require an address—a memory location—in decimal that follows the instruction. This address tells BASIC where to direct the action of the command. The address may be specified as a positive number or as a negative number. The positive version of the address is in the range 0 to 65535 possible addresses—the total address space of the Apple. (The 64K aux memory in 128K Apple //e or //c computers cannot be directly accessed from BASIC.) The negative version of the address is in the range -1 to -32767 and represents the addresses from 65535 to 32768. The addresses from 0 to 32766 are always expressed as positive numbers. Keep in mind that the same address can be expressed both ways: -936 is identical to 64600.

This dual method of representing addresses is a holdover from Integer BASIC, which can't deal with numbers larger than 32767; the negative version of an address allowed Integer BASIC programmers to access memory locations above 32767.

PEEK directs BASIC to *fetch* the value stored in a memory location and is often used in conjunction with the print statement. PRINT PEEK(64435) or PRINT PEEK(-1101) will print a 6 if you're using a //e or //c computer and a 234 if you're using an Apple IIplus or II with an autostart monitor.

POKE directs BASIC to *store* a value in a memory location. In addition to the address, you must specify what value you want to store. POKE 230,32 will store the number 32 in the memory location 230. You can't store a value in a ROM memory location.

CALL directs BASIC to *execute* a machine language subroutine that starts at the address specified. CALL -936 or CALL 64600 clears the text screen. (Just like HOME in Applesoft.)

Bug Alert. These instructions can be dangerous. By storing an incorrect value in a memory location or CALLing a memory location that's not a subroutine, you can cause a system crash and may even destroy the data on your disk.

How do I list a program to the printer?

With a printed listing in hand, you can leave the computer, pour a cup of coffee or tea, and track down whatever bugs may have crept into your program. You may also want to keep printer listings with finished projects to help you document what you've done.

Printers are normally connected to slot 1. (See What's a slot? p. 12) If yours is not, change the 1 in the following examples to the appropriate slot number.

LOAD the program you wish to list.

Activate the printer with PR#1 RETURN.

List the program by typing LIST RETURN.

Deactivate the printer with PR#0 RETURN or PR#3 RETURN. Use PR#0 if you are working on the 40-column screen and PR#3 if you are working on the 80-column screen.

If a printer is not connected or if the connection to the printer is not correct, the computer will "hang," that is, it will appear to be totally unresponsive. Actually, the Apple thinks it is talking to a printer and is not smart enough to know better. To recover, press CONTROL RESET.

Don't forget to deactivate the printer. If you do not, everything you do that would normally be printed on the Apple screen will also be printed on the printer.

With many printers, you'll see the typical Apple 40-column listing. If you'd rather use the full width of the computer paper, type CONTROL I 80N after the PR#1 but before typing LIST. This command is directed at the printer interface and may not work on your system. If it does not work, consult the documentation that came with your printer and printer interface card.

This method will also enable you to get a printed CATALOG. Follow the steps above, but instead of LIST, type CATALOG. This is especially handy in ProDOS if you are using a color monitor. You can get the full 80-column wide CATALOG on the printer instead of the shorter CAT that you must use on your CRT.

How do I use a printer in a program?

Using the printer when you are running a program is much like using DOS in a program. The printer is activated with a PRINT CHR$(4);“PR#1” command in the program. Of course, you can use any program line number and must change the 1 to the number of the printer slot.

Once activated, everything that would normally be printed on the Apple screen will be printed on the printer.

To deactivate the printer, include the following line in your program: PRINT CHR$(4);“PR#0”. Change the 0 to a 3 if you are working on the 80-column screen.

If your program uses the TAB function to position text, it may not work properly on the printer. Replace TAB with the SPC function or consult the documentation that came with your printer or printer interface card.

If you have a printer that is capable of graphics and a printer interface card that can decode the Apple screen into graphics commands, you can use the same technique to print the current Apple graphics screen. (See How do I print graphics? p. 170)

The following program creates a graphic grid, then dumps it to the printer. This program will work only with an Orange Micro Grappler card and an Epson or other compatible printer.

```
10 HGR: HCOLOR=3
20 FOR X = 0 TO 279 STEP 4
30 HPLOT X,0 TO X,191
40 NEXT X
50 FOR Y = 0 TO 191 STEP 2
60 HPLOT 0,Y TO 279,Y
70 NEXT Y
80 PRINT CHR$(4);"PR#1": REM activate printer
90 PRINT CHR$(9);"G": REM dump hi-res page 1
100 PRINT CHR$(4);"PR#0": REM deactivate printer
```

To make the program work with the Interactive Structure's Pkaso interface, change line 90 to read PRINT CHR$(9);“H”.

How can I control a program from the keyboard?

The most obvious ways to accept input from the keyboard—the Applesoft INPUT and GET commands—also stop the program. Many graphics programs benefit from an interrupt, the ability to read the keyboard without stopping the program.

The simplest way to accomplish this is to poll the keyboard. Polling is a programming technique where the program continually looks to see if something happened—most often a change in the status of a peripheral, such as a graphics input device. (Was a key or a button pressed?) If the status is unchanged, the program goes its merry way; if the status is changed, the program branches to deal with the action.

There are two steps in polling the Apple keyboard. First, the keyboard must be cleared of previous keypresses. A simple POKE accomplishes this. Then, the keyboard must be read using a PEEK. When the value of the I/O location being PEEKed is greater than 128, it means a key has been pressed and the desired action is effected. Here's all of this in action:

```
10 POKE -16368,0:REM CLEAR KEYBOARD
20 X=PEEK(-16384):IF X<128 THEN GOTO 20
30 X=X-128:X$=CHR$(X)
40 IF X<32 THEN X$="CONTROL "+CHR$(ASC(X$)+64)
50 PRINT X$;" KEY WAS PRESSED"
60 IF X=27 THEN END
70 GOTO 10
```

Try this program with the arrow keys! The keypress values represent the ASCII value of the keys. Note that ASCII 27 is the ESC key and, thanks to line 60, will end the program. Once you've captured the ASCII value of a key, you can do whatever you want to do with this value. Many paint programs, for example, use single keypresses to represent commands—if you press C when you use one program it clears the screen, while with another program, pressing C changes the paint color.

How can I control a program with paddles?

Polling—the same technique used to control a program from the keyboard—can be also used to control a program with paddles. Paddles or their analogs (joysticks and touch tablets) offer four means of input: paddle 0, paddle 1, button 0, or button 1. The response of the paddle inputs is different from the button inputs.

Paddles affect memory locations -16284 (49252) to -16281 (49255). Applesoft BASIC supports direct reading of the paddles via the PDL() function. X= PDL(n) assigns the value of paddle n to the variable X. These values can be used to determine X and Y plotting positions. Here's a sketching program to illustrate the paddle inputs:

```
10 HGR:POKE -16302,0
20 HCOLOR=3
30 X=PDL(0):Y=PDL(1)
40 IF Y>191 THEN Y=191
50 T=T+1:IF T=1 THEN HPLOT X,Y
60 HPLOT TO X,Y
70 GOTO 30
```

Run this program and you'll see that the paddles don't let you get to the far right of the screen. This is because paddles only report a limited range of numbers to BASIC (from 0 to 255) and Apple's hi-res screen has a horizontal (X) range from 0 to 279. A little mathematical computation can compensate for this limitation. We'll work with X coordinates from 0 to 139 since the color resolution of the Apple is only 140 × 192. These modifications will give you access to the whole screen. (The 1.8 in line 30 is the result of dividing 255 by 140.)

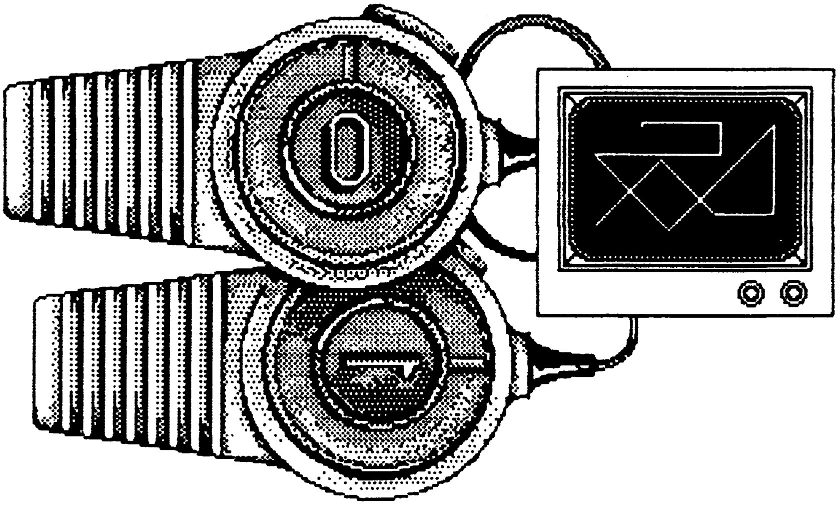
```
30 X=PDL(0)/1.8:Y=PDL(1)
35 X=X*2:IF X>279 THEN X=279
```

The buttons are more difficult to work with as BASIC doesn't directly support them. A PEEK, however, can determine if a button has been pressed. Add this line to the sketch program:

```
65 IF PEEK(-16287)> 127 THEN POKE -16301,0:STOP
```

PEEKing location -16287 (49249) reads button 0. Button 1 can be read by PEEKing location -16286 (49250). When either PEEK yields a value greater than 127, the button has been pressed.

By the way, custom-wired paddles can access paddle 2, paddle 3, and button 2, but none of the commonly available commercial paddles or joysticks access these inputs.



How can I control a program with the mouse?

This information applies only to the Apple Mouse II.

The Apple Mouse II plugs directly into the game port of the Apple //c but requires an interface card plugged into slot 4 of an Apple II, IIplus, or //e computer. The game port of the //c and the mouse interface card have extensive ROM programming to support mouse activities. Using the mouse is like using any other Apple peripheral that plugs into a slot.

A device's ROM is turned on by printing an initialization command to the correct slot—information is acquired from the device by an input command from that slot. Once initialized, the mouse produces three string values: an X location, a Y location, and a status. When you've finished using the mouse, you must turn it off and return to normal I/O.

The following program goes through the necessary steps. Press ESC to quit the program.

```
10 HOME: SLOT=4:REM CHANGE THIS IF NOT IN SLOT 4
20 PRINT CHR$(4);"PR#";SLOT
30 PRINT CHR$(1):REM INITIALIZE MOUSE
40 PRINT CHR$(4);"PR#0"
50 PRINT CHR$(4);"IN#";SLOT
60 INPUT X$,Y$,S$
70 VTAB 10:HTAB 10:PRINT X$,Y$,S$
80 IF PEEK(-16384)=155 THEN POKE -16368,0:GOTO 200
90 GOTO 50
200 PRINT CHR$(4);"IN#0": PRINT CHR$(4);"PR#";SLOT
220 PRINT CHR$(0):REM TURN OFF MOUSE
230 PRINT CHR$(4);"PR#0"
```

Deciding what to do with the mouse values and determining how to correctly use the information is tricky. The mouse returns X and Y coordinates in the range of 0 to 1023. The BASIC VAL() function can be used to convert these coordinates to numeric information (X=VAL(X$)). The mouse location where the CHR$(1) command was received is 0,0. The status information (S$) is a three-character string. The first character is "+" if no key was pressed and "-" if a key was pressed. The second character of the status information is 0. The third character is 1 if the button is pressed, 2 if the button was just pressed, 3 if the button is released, or 4 if the button was just released.

What's HIMEM and LOMEM?

You don't have to concern yourself with HIMEM and LOMEM unless you're writing large BASIC programs and using high-resolution graphics. If you start getting OUT OF MEMORY errors, or strange things happen to your images, then it's time to manipulate Apple's memory.

The BASIC statements HIMEM: and LOMEM: allow you to set aside safe areas of memory for assembly language routines and high-resolution graphics.

LOMEM

BASIC programs are stored in memory location 2048 and up. As the program runs, it uses more memory to store variables and arrays. The memory used for this purpose starts at a location called LOMEM. As the program gets longer, the value of LOMEM increases. Eventually, the top of the program or your variables can collide with page 1 of hi-res graphics.

If your variables are being stored on hi-res page 1, you'll see random dots on the screen and all variables will be set to zero when you issue a HGR instruction. In this case, change LOMEM.

If you're using only page 1 of hi-res graphics you could move LOMEM to location 16384 by using the BASIC command LOMEM: 16384 at the beginning of your program before any variables are defined. Now, variables and arrays will be stored above hi-res page 1. If you're using page 2 of hi-res graphics, move LOMEM to location 24576.

If your program itself is colliding with hi-res page 1, the end of the program will disappear when you issue a HGR command. In this case, change the start-of-program. (See How do I relocate programs? p. 102)

HIMEM

HIMEM problems occur when you are using special graphics routines like PICDRAW from The Graphics Magician and HRCG from the DOS Toolkit. Because these subprograms sit just below DOS, you must move the place where BASIC stores it's string variables by using a HIMEM: command.

If your program is short and doesn't use many string variables, you may want to collapse BASIC's workspace to 6K by including a HIMEM:8192 in your program. You now have most of memory to store graphics data.

How do I relocate programs?

Often, especially with graphics, it's necessary to relocate a program. When would you need to do this? If your program is long and it, or its variables, disappear when you issue an HGR command, your program is stretching into hi-res memory. The HGR command puts zeros in memory from 8192 to 16283, wiping out your program, its variables, or both, if they get in the way.

Normally, Applesoft programs start at location 2048 and extend upward in memory. As it executes the program, BASIC stores variables and variable pointers—only strings are stored in a different location. Normal string variables such as A$="Hello" are stored *within* your program. Manipulated string variables such as A$=A$+ "there, Joan." are stored starting at HIMEM downward. Look at the Applesoft memory map and you'll see that HIMEM is normally immediately below DOS.

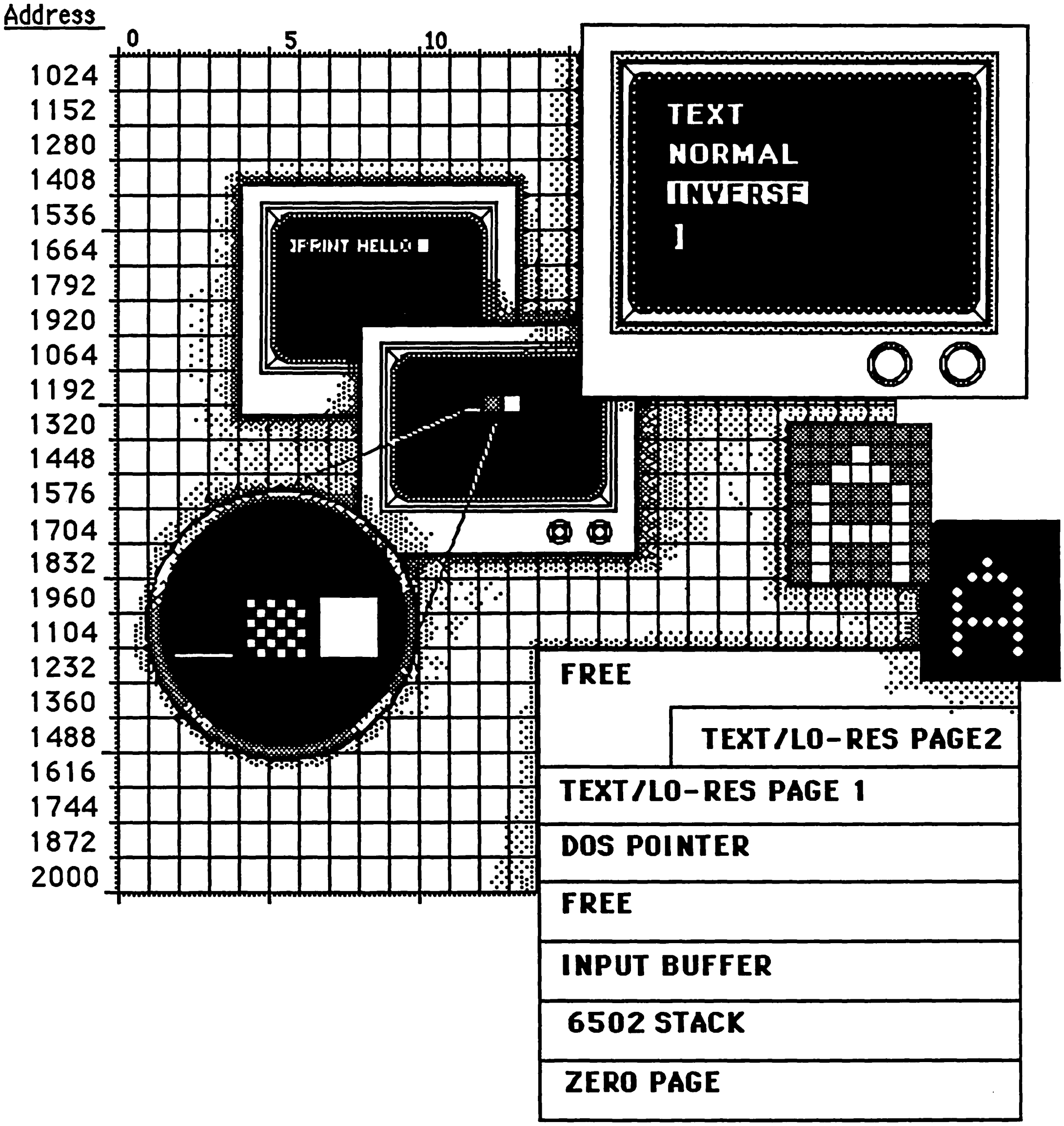
Without graphics, then, only very long programs run out of memory (the program collides with string variables). With graphics however, you only have 6K for a program and its variables (8192 -2048=6144). Long graphics programs are, therefore, better started at 16384 or 24576, depending on whether they use page 1, page 2, or both hi-res pages.

The place where your program starts is stored in locations 103 and 104. (The command PRINT PEEK(103)+PEEK(104)*256 should print 2049.) To move a program we alter these locations *prior* to LOADing or writing the program. The key is *prior*.

Once these locations have been changed, Applesoft loses track of the program; only one DOS command can follow the relocation instructions. Therefore, you must create a preprogram which relocates the start-of-program pointers (103 and 104) and then RUNs your *real* program. Enter this preprogram and **SAVE it before you RUN it!**

```
10 LOC=24576
20 HA = LOC/256
30 LA = LOC - HA/256
40 POKE LOC-1,0:POKE 103,LA:POKE 104,HA
50 PRINT CHR$(4);"RUN your program"
```

Change "your program" to the name of a program on your disk or you'll get a FILE NOT FOUND error. Even if you get this error, the start-of-program pointers will be moved.


Address
0
5
10
1024
1152
1280
1408
1536
1664
1792
1920
1064
1192
1320
1448
1576
1704
1832
1960
1104
1232
1360
1488
1616
1744
1872
2000
TEXT
NORMAL
INVERSE
]
]PRINT HELLO
FREE
TEXT/LO-RES PAGE2
TEXT/LO-RES PAGE 1
DOS POINTER
FREE
INPUT BUFFER
6502 STACK
ZERO PAGE

About Text

About text

What's text mode?

The Apple has a memory-mapped display, that is, the image shown on the monitor or TV is an active part of the computer's main memory. The video circuitry is wired to sweep through an area of memory 60 times a second and construct a video signal from what it finds there. There are four chunks of memory that the video circuitry can look at.

As the video circuitry scans memory, it interprets it. The information in two chunks of memory, the first and second text pages, is interpreted as *characters* to be displayed whereas the information in another two chunks of memory, the hi-res graphics screens, are interpreted to be *pixels.* A glance at the Apple memory map reveals that these are located in different parts of memory. (See What's a memory map? p. 39) The primary text screen, for example, begins at location 1024 and extends to 2047, while the primary graphics screen begins at location 8192 and extends to 16383.

When the Apple is directed to display text, with the TEXT instruction, the computer can only display the character images stored in the character generator ROM—one image for each possible byte value: 0 to 255.

Can I use graphics in text mode?

Yes and no. Aside from the four-line text window at the bottom of the screen, you can't mix the Apple's lo-res, hi-res, or double hi-res graphics with text.

You can, however, use the Apple's text characters to create graphics on the text screen. The effect is similar to typewriter art. Many novice programmers begin designing computer games using the text screen. We have seen rather effective simulations of such popular games as PacMan, Hangman, and so on—all in text mode. These become more effective by judicious use of Applesoft's INVERSE and FLASH commands.

Apple //c and enhanced Apple //e computers can display a selection of graphic characters called MouseText. These characters were incorporated into the Apple character set in place of inverse characters and they can be used to create borders and other graphics on the text page. (See How do I use MouseText? p. 118)

What's ASCII code?

ASCII is an acronymn for American Standard Code for Information Interchange and represents the way most computers store text information.

This standard defines 128 characters, each of which has been assigned a number from 0 to 127. The upper-case "A" is 65 in ASCII code and the lower-case "a" is 97. All the symbols used in written communication are included in the ASCII set as well as a few that are used only in computer work.

The following program displays most of the ASCII set as it is stored in the Apple's character generator ROM:

```
10 HOME
20 FOR I = 32 TO 127
30 PRINT I; SPC(2+(I<100)); CHR$(I); SPC(2) ;
40 NEXT I
```

Even though the code is standarized, you may not see the full set. If you're using an Apple II or Apple IIplus without a modified character generator, characters 97 to 122 (the lower-case characters) will repeat the upper-case alphabet found from 65 to 91. This is how the character generator was programmed to deal with these characters; a printer would print the correct lower-case representation.

The characters numbered from 0 to 31 are control codes. They can't be shown on the screen because they have no physical form but they do have functions. Character 7, for example, beeps the speaker. Try it: type PRINT CHR$(7) RETURN or type the code directly by pressing CONTROL G. The key CONTROL modifies G to produce an ASCII code of 7 rather than 71 (upper-case G) or 103 (lower-case g).

Computers store character information in bytes, and a byte can store up to 256 numbers, so each computer manufacturer has an extended code that defines the characters from 128 to 255. Apple uses the extended (full-byte) code to generate inverse and flashing characters.

In the Apple //c and enhanced Apple //e computers, some of these extra characters have been defined as tiny symbols rather than letters or numbers. Apple calls the symbols "MouseText" because they're most often used in conjuction with the AppleMouse.

Why don't ASCII characters display correctly?

The ASCII code is not the same as the screen display code. In general, you must POKE 128 plus the ASCII value of the character to be displayed into screen memory. As to why...

The original Apple character generator ROM was small—too small to contain a full ASCII character set. When a couple of young guys are designing a computer, they have to consider costs. Because there wasn't a full character set, Apple decided to use the unused codes to generate flashing and inverse characters. Character generator ROMs got cheaper and cheaper and when the Apple //e was introduced, Apple could afford to put a full character generator ROM into the computer. When the //c was released, Apple splurged, it even provided graphic characters and soon upgraded the //e to be compatible with the //c. So much for history.

No Apple computer displays what you would expect from the ASCII code. If you POKE 65 into text screen memory, you would expect to get a capital A. Instead, the A flashes. To get a nonflashing A, you'd have to POKE 65 + 128 into screen memory.

The translation of ASCII to display codes is an unfortunate complication of the Apple text screen. Fortunately, BASIC provides us with more than adequate support for text screen printing, and only assembly language programmers need worry about the details.

The following table details the codes used to display the characters in the Apple's ROM character set:

CODE	CHARACTERS	FORMAT
0-31	upper-case letters	inverse
32-63	special chars[1]	inverse
64-95	upper-case letters	flashing | special[2]
96-127	special chars	flashing | inverse lower-case
128-159	upper-case letters	normal
160-191	special chars	normal
192-223	upper-case letters	normal
224-255	lower-case letters[3]	normal

[1] Special characters include punctuation and numbers.

[2] Codes 64-95 display as inverse upper-case on old //e or, if enabled, as MouseText on //c and enhanced //e.

[3] On II and IIplus these display as punctuation and other characters.

What's the difference between 40- and 80- column mode?

The standard Apple text page displays 24 lines of 40 characters: a total of 960 characters. A 40 character display is inadequate for word processing and other business applications.

80-column cards are peripherals that double the number of characters on each line, producing a display with 24 rows of 80 characters. As you might suspect, the wider display takes twice the memory of the 40-column display.

There's a wide variety of 80-column peripherals for the Apple II and Apple IIplus computers. These cards, available from several manufacturers, plug into slot 3 of the Apple.

The Apple //e was designed with a special slot—the Aux slot—for an 80-column card, although most of the cards designed for the Apple II and IIplus computers will also work in slot 3. There's a compelling reason to choose Apple's own extended 80-column card (or one that's compatable with it) for the Apple //e. The reason lies in the way this card works in comparison to the other type of card.

Slot 3-type cards contain the entire 80-column screen in memory on the card. Because Apple screen memory is not used for the display, these cards have better character sets and they can display characters in foreign languages without changing Apple's built-in ROM character generator. These 80-column cards, however, don't have enough memory nor do they have the necessary software support to display graphics. All graphics must be done with main, motherboard memory.

Aux slot-type cards, on the other hand, have special video circuitry and software to shuffle memory so that half the 80-column display is on the motherboard and half is on the card. The display is interleaved: aux memory, main memory, aux memory, main memory, and so on across the screen. While more complicated, this allows for extended graphics modes: double-hi-res and double-lo-res graphics. At the time of this writing, these extended modes are not available on standard slot 3-type 80-column cards.

Apple //c computers come with the Apple extended 80-column card built in.

How do I switch between 40- and 80-column modes?

If there's an 80-column card in the Apple II or Apple IIplus computer that you're using, consult the manual that came with the card. Most cards are activated with a PR#3 but are deactivated with varing commands.

Apple //e and //c computers have "official" 80-column modes. As far as text is concerned, the Apple extended 80-column card operates identically to the Apple 80-column card.

The Apple 80-column card features three text display modes: regular 40-column, 80-column, and 80-column-active, 40-column-display. This last, tongue twisting mode is for use with a television or composite monitor; it gives you the benefits of 80-column mode but with the 40-column text display.

Activate the Apple 80-column card in an Apple //e computer by typing PR#3. The screen will instantly change to the wider mode and the cursor will change from a flashing checkerboard to a non-flashing white rectangle. If you're not using a monochrome monitor or RGB color display, you may not be able to read the 80-column screen. Switch to the 80-column-active, 40-column-display mode by keying ESC 4. The white cursor will remain, but the screen will again be forty columns wide.

This mode offers the "escape mode" cursor, a good screen editing option. When you press ESC, the cursor changes to an inverse plus sign indicating that you are in escape mode and the arrow keys can move the cursor around the screen. Return to the 80-column mode by keying ESC 8 . To return to regular 40-column mode, type ESC CONTROL Q. Do not type PR#0!

Most of these instructions apply to the Apple //c computers, but you don't have to type PR#3 to activate the 80-column card—simply press ESC . ESC 4 takes you to 80-column-active, 40-column mode, ESC 8 takes you to 80-column mode, and ESC CONTROL Q returns you to standard Apple 40-column mode. On this computer, the escape mode cursor is always an inverse plus, regardless of screen mode.

If there's no 80-column card in slot 3, PR#3 will lock up the computer; press CONTROL RESET to recover the system.

Can I get lower-case on the Apple II or IIplus?

A significant difference between the Apple II and IIplus computers and the Apple //e and //c computers is in the keyboard, character generator, and system text support. The newer Apples have full upper- and lower-case text ability; the older Apples do not.

Although the Apple's native character generator cannot display lower-case characters on the text screen, a software character generator can display them on the graphics screens. Many word processors and graphics programs allow you to type upper- and lower-case letters using ESC as a shift key.

For those who need true upper- and lower-case display, a kit can be purchased. This kit allows you or a dealer to replace the standard Apple character generator with one that has a full upper- and lower-case character set.

Unfortunately, installing this kit will not give you a functioning shift key. A second modification is necessary to activate that key. This entails connecting the keyboard to one of the inputs on the game port. Most Apple II and IIplus software that would benefit from a true upper- and lower-case display, such as word processors, are programmed to take advantage of this modification.

Another option is to add a "keyboard enhancer" to transform the keyboard into a full upper- and lower-case keyboard. Keyboard enhancers don't require special programming to use the shift key.

How do I go from upper-case to upper- and lower-case?

This answer to this question depends upon the type of Apple you're using.

If you're using an Apple II or IIplus computer without keyboard or character generator ROM enhancements, your keyboard can only generate upper-case characters. (See Can I get upper- and lower-case on an Apple II or IIplus? p. 110) Programs that display lower-case are doing so on the hi-res graphic screen by using a RAM character generator. Most of these programs convert your typing to lower-case, capitalizing only when ESC has been pressed.

If you're using an Apple II or IIplus with an upgraded ROM character generator or keyboard, you can type full upper- and lower-case text; assuming the program you're running can handle it. DOS 3.3 and Applesoft BASIC are two programs that require upper-case only input. ProDOS accepts upper- and lower-case input.

If you're using an Apple //e or //c computer, CAPS LOCK toggles or switches between upper-lower-case and upper-case-only modes. When you're running old software—like DOS 3.3, Applesoft and many graphics programs—that has not been modified to accept lower-case input, you must keep CAPS LOCK in the upper-case only positon. Most programs that allow you to type on the hi-res screen will accept both upper- and lower-case input and you can release CAPS LOCK when you're typing. However, when you return to a menu or issue a program command, chances are it will require upper-case input.

Many times a program that reads upper-case only will instruct you to type a letter such as P for Paint and you'll keep pressing the P key to no avail. Before you decide the program isn't working, try pressing CAPS LOCK. You might have inadvertently hit it and gone into lower-case mode.

How do I position text on the screen?

Applesoft supports several methods of positioning text.

When you use the PRINT statement, BASIC normally goes to a new line (performs a carriage return) after executing the command. You can change this. Adding a semicolon or a comma after the information to be printed, affects the way in which the information is displayed. Try the following program:

```
10 PRINT "HELLO"
20 PRINT "THERE"
30 PRINT "HELLO";
40 PRINT "THERE"
50 PRINT "HELLO",
60 PRINT "THERE"
```

Run this program to see the three variants of the PRINT statement. The comma is a tab indication. On the 40-column screen, there are 3 tab stops before BASIC moves to the next line; on the Apple 80-column screen, there are 5 tab stops. The semicolon stops BASIC from going to a new line.

There are four instructions for text positioning: TAB(n), SPC(n), VTAB n, and HTAB n. Each requires a number in place of the "n."

TAB(n) directs printing to the screen column specified by n. TAB is used within a PRINT statement—PRINT TAB(20);"HELLO"

SPC is a space over function. SPC(n) skips the number of spaces indicated by n. SPC is also used in a PRINT statement—PRINT SPC(20);"HELLO".

VTAB n directs printing to the vertical screen row specified. VTAB 1 is the first row of the screen and VTAB 24 is the last row. VTAB is a statement all by itself—VTAB 10: PRINT "HELLO".

HTAB n moves to the horizontal screen column specified. HTAB 1 is the leftmost column of the screen and HTAB 40 or HTAB 80 is the rightmost column of the screen. HTAB, like VTAB is used alone—HTAB 10: PRINT "GOODBYE".

Bug alert. You cannot use these instructions to print in the lower right-hand position of text screen (VTAB 24:HTAB 40); the screen will scroll. Instead, POKE the character's ASCII value (plus 128) into location 2039. For example: POKE 2039,65+128.

How do I center text on the screen?

Centering text is useful when designing title screens. When the text is centered, the screen has a balanced and professional appearance.

Centering text on the computer screen is the same as centering text on a typewritten page. Count the number of columns available, subtract from that number the total number of characters in the word or phrase to be centered, and divide the difference by two. The resulting number tells you in which column to place the first character of the word or words to be centered.

Because you're working with a computer, there's no need to be bothered with this arithmetic. Instead, use the following centering algorithm and let Apple do the computing:

```
HTAB (CT-LEN (A$) ) / 2
```

HTAB is the number of spaces indented from the left side of the screen. CT is the number of columns on the screen. This number will be 40 or 80, depending upon your particular Apple's display. LEN is BASIC's length function which, in this case, will give the number of characters contained in the string (A$) that is enclosed by parenthesis. A$ is the word or phrase that you want centered.

Try this:

```
10 TEXT: HOME
20 CT=40
30 A$ = "IT WORKS!"
40 HTAB (CT-LEN (A$) ) / 2
50 PRINT A$
```

When you want a number of words or phrases centered on the screen, use the centering algorithm as a subroutine. (See What's a subroutine? p. 91)

Here's an algorithm that will right justify text. It 's used the same way as the centering algorithm.

```
HTAB CT - LEN (A$)
```

What's a text window?

Apple's normal text display is 24 rows of 40 or 80 characters. Sometimes, you may want your text displayed within a smaller "window." The left, right, top, and bottom sides of the window can be controlled by using POKE statements to memory locations 32 through 35. When these locations are changed, the size of the text window changes.

Memory Location	Window Control	Legal Range
32	left margin	0- 39
33	right margin	1- 80 (minus left margin)
34	top margin	0 to bottom line
35	bottom margin	top line minus 24

The syntax is POKE XX,n where "XX"is the memory location and "n" can be any number within the legal range: for example POKE 33,20 will limit the number of characters on a line to 20. POKE 35,18 sets the bottom margin to 18.

Setting a text window doesn't clear the text screen or put the cursor into the window. If you have text on the screen, set a window, and then type HOME, only the area within the window you set will be cleared. To move the cursor use HTAB and VTAB. Try this:

```
10 INVERSE: WI=40: REM WI=80 for 80 column screen
20 FOR I = 1 TO 24:FOR J = 1 TO WI-(I=24)
30 PRINT ".";: NEXT J: NEXT I: POKE 2039,46
40 POKE 32,10: REM left margin at 10
50 POKE 33,15: REM 15 characters wide
60 POKE 34,5: REM top margin at 5
70 POKE 35,15: REM bottom margin at 15
80 NORMAL: HOME
90 PRINT "THE TEXT WINDOW"
100 VTAB 15: PRINT "PRESS RETURN";
110 INPUT A$
120 TEXT: REM restore full text window
```

Applesoft programmers sometimes move the right margin to 33 or 73 for easier program editing. Applesoft will not put extra spaces into the program with the reduced window.

Bug Alert. Never set the value of the bottom of the window (POKE 35,n) smaller than the value of the top of the window (POKE 34,n). Never set the value of the right margin (POKE 33,n) to 0 or larger than the screen (40 or 80 column) you are using. Applesoft will crash and you'll have to reboot.

Can the Apple display text in color?

The standard Apple text display is monochromatic (black and the color of your CRT phosphors). Some RGB monitors support colored text, but the quality of the display depends upon the particular monitor and its interface card.

The Apple RGB extended 80-column card for the Apple //e (or equivalent) allows you to display text in 16 colors.

You can create colored text screens using hi-res graphics.

How can I enhance the text display?

To spice up the Apple's text screen, you can display text as inverse (black on white, green, or amber) or flashing (fast flips between normal and inverse text).

Try the following:

```
10 HOME
20 PRINT "HELLO"
30 INVERSE: PRINT "HELLO": NORMAL
40 FLASH: PRINT "HELLO": NORMAL
```

Judicious use of INVERSE text can enhance the text display, but be discriminating in your use of FLASHing text. It can be very disconcerting (if not dizzying) to see a screen filled with flashing characters.

Flashing lower-case is not available—the letters will display as punctuation marks and other symbols. Furthermore, FLASH will not work when the Apple //e and //c computer have active 80-column cards. (That is, when the cursor is a solid rectangle.)

Inverse also behaves differently on different Apples. On the Apple //c and the enhanced //e computers, the commands INVERSE:HOME will clear the screen to white (or green or amber). Older Apples will resolutely maintain a black screen.

Note that the Applesoft command, NORMAL, returns the display to regular text regardless of whether the previous instruction was FLASH or INVERSE.

Why is the text display fuzzy and fringed with color?

Apples were designed to allow text display on television screens. Considering the resolution of a standard television, 40-column text is a good balance between the number of characters displayed and readablity. On inexpensive TV sets or composite color monitors, however, even 40-column text is fuzzy. The only solution is a monochrome or RGB monitor. (See Do I need a monochrome monitor if I work in color? p. 19)

As for the color fringing, the Apple's TV signal is not a clean signal. Without special circuitry to flatten the color component of this signal, the text has colored fringes. The earliest Apple II computers didn't have such a circuit. One of the first modifications of the Apple motherboard was to include a "color killer" circuit.

Most Apple II and IIplus computers have this circuit; all Apple //e and //c computers have it. When the computer is in text mode, the circuit kicks in and eliminates the color component of the television signal. It appears as if Apple has redesigned this circuit several times, as it works better on the Apple //e and //c than on the older Apples.

If the color killer doesn't work on the Apple you're using and your dealer can't repair it, consider purchasing an inexpensive monochrome monitor. The color component of the signal will be ignored and your text and graphics will be crystal clear. A television can remain connected to the RF connector for color display. If you're already using a composite monitor, an inexpensive Y-jack will enable you to connect both the color and the monochrome monitor to the video output jack at the rear of the Apple.

How do I use MouseText?

If you are using an Apple //c or enhanced //e, you have MouseText, a set of small graphic characters that can be displayed on the text screen.

The MouseText characters begin with display character number 64 and extend for 32 characters. These characters can either be PRINTed or POKEd onto the text screen.

Here's a program that shows the MouseText characters.

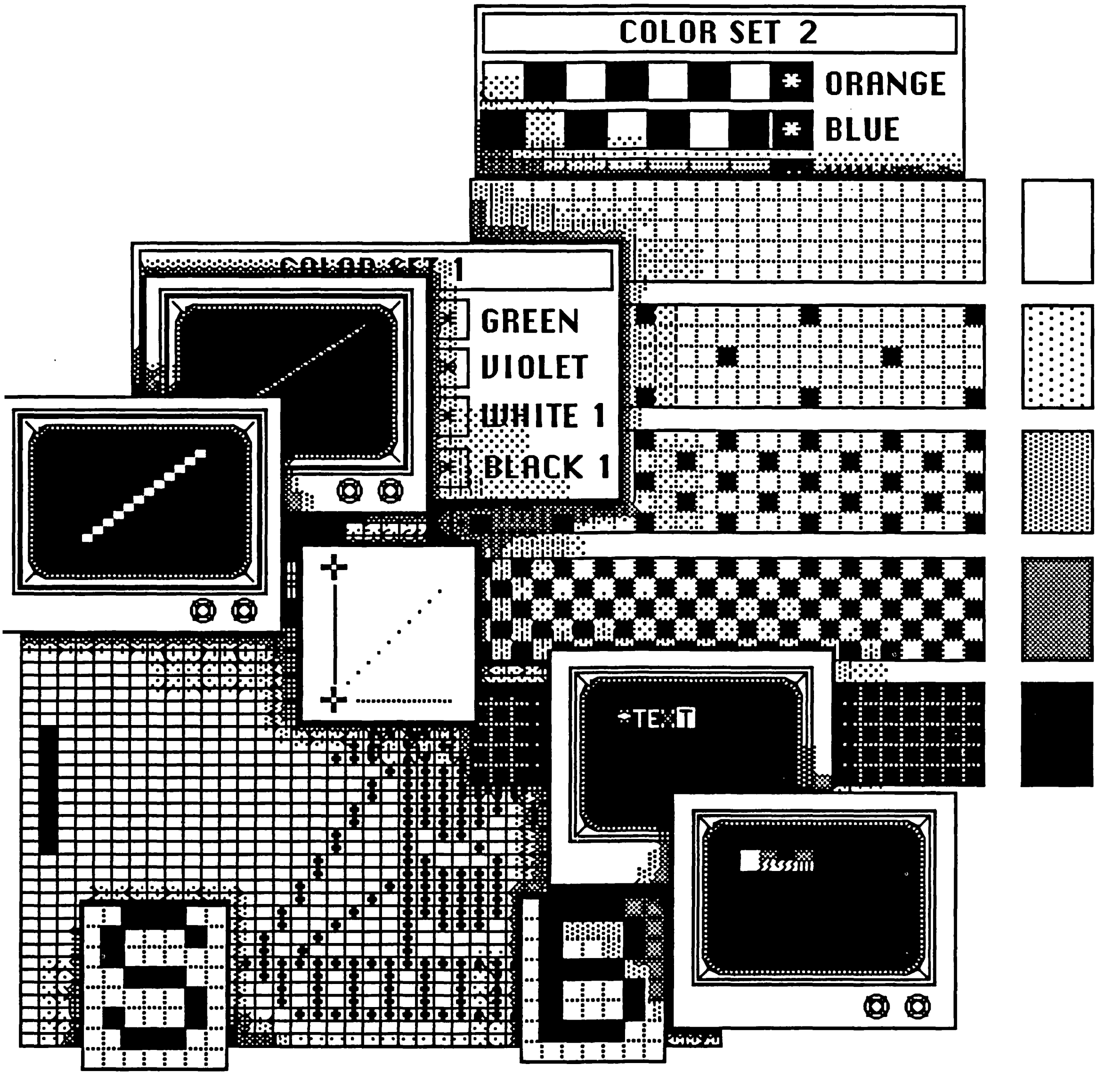
```
10 HOME
20 PRINT CHR$(4)"PR#3"
30 PRINT CHR$(27);
40 PRINT CHR$(15);"@ABCDEFGHIJKLMNOPQRSTUVWXYZ[|]^_";
50 PRINT CHR$(14);CHR$(24)
```

Here's a similar program using POKE:

```
10 HOME
20 FOR I = 0 TO 26 :POKE 1024+I*2,I+64: NEXT I
```

Finding the correct place in which to POKE the MouseText characters is a chore due to the baroque way the Apple screen is mapped. (See Why do pictures load like venetian blinds? p. 164) First you must determine the location of the leftmost byte of the screen row on which the character is to appear. The simplest way to do this is look it up on a screen map. (See the Appendix p.283) Add to this address the horizontal position of the character and POKE the character value into the address. It's not a bad as it sounds—here's an example:

```
10 HOME
20 FOR I = 1024 TO 1024+39:POKE I,159: NEXT I
30 POKE 1152,95: POKE 1152+39,90
40 POKE 1280,95: POKE 1280+39,90
50 POKE 1408,95: POKE 1408+39,90
60 FOR I = 1536 TO 1536+39:POKE I,76: NEXT I
70 VTAB 3: M$="My Programs": GOSUB 500
80 POKE 1280+(H-3),64
90 POKE 1280+(H+LEN(M$)),64
100 END
500 H=INT(20-LEN(M$)/2)
510 HTAB H:PRINT M$: RETURN
```



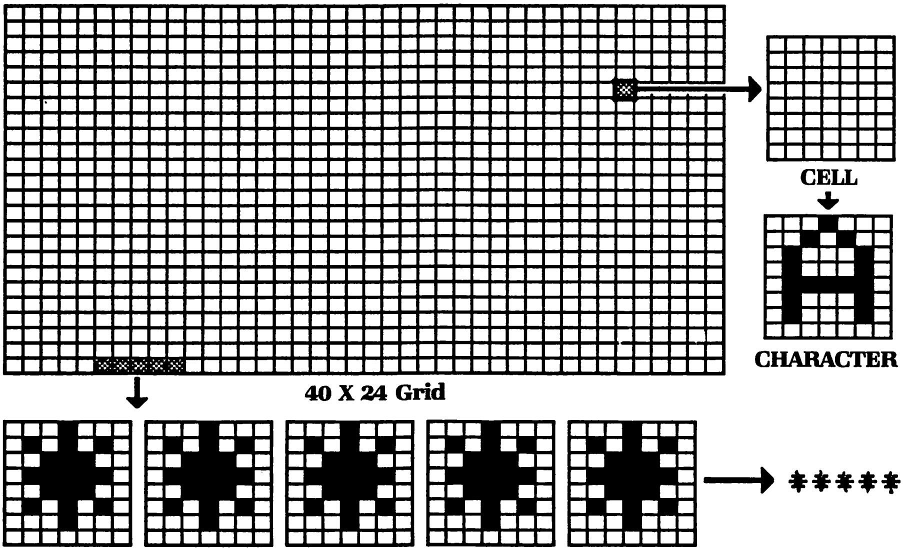
About graphics

About graphics

What's a computer graphic?

A computer graphic is a computer-generated image created for visual impact, usually to communicate an idea. Graphics can be created in all Apple display modes.

Text mode offers you control of a 40 × 24 (or 80 × 24) grid of characters. Imagine this grid as a sheet of graph paper; any one of approximately 96 different characters can be placed into each "cell" of this grid. Each character is made up of seven horizontal dots and eight vertical dots. The actual letters are 5 × 7, leaving blank one vertical row of dots on each side and one horizontal row on the bottom. Although you can't control individual dots—the text characters have been predefined for you by Apple—the characters can be used to create graphic designs and borders. The computer manipulates the screen as if it contained meaningful text.



The graphics modes, on the other hand, allow you to control each dot on the screen. In these modes, the screen is dealt with as a grid of dots. The dots may have attributes, like color, but of themselves they're only dots.

In graphics mode, the graphic is created by combining the dots to form the desired image (a row of dots creates a line). Each of these dots is a pixel, which is short for picture element.

What's resolution?

Resolution refers to the number of pixels that can be displayed on a screen.

Apple's low-resolution mode (lo-res) provides 40 horizontal and 40 vertical pixels (1,600 total) without the text window or 40 × 48 pixels (1,920 total) with the text window. The Apple //e (with 128K) and //c computers have a double-lo-res mode with double the number of horizontal pixels, for a total of 3,200 pixels with the text window, or 3,840 without the text window.

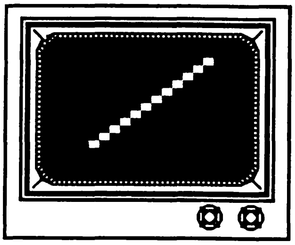
In lo-res modes, diagonal lines appear extremely jagged or, as the computer graphics professional would say, aliased. (See What's aliasing? p. 123)

All of Apple's high-resolution modes share a vertical resolution of 192 pixels. Horizontally, hi-res mode has 280 pixels, for a total of 53,760 pixels. The Apple //e (with 128K) and //c computers have a double-hi-res mode with 560 pixels on the horizontal axis, for a total of 107,520 pixels on the screen.

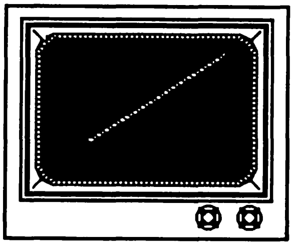
With higher resolution there's less aliasing, and greater detail can be put into computer graphics.

This program shows the difference between lo- and hi-res:

```
10  HOME: GR
20  COLOR=13
30  FOR I = 0 TO 39
40  PLOT I,I: NEXT I
50  VTAB 21: INPUT "PRESS RETURN FOR HI-RES LINE";A$
60  HOME: HGR
70  HCOLOR=3
80  FOR I = 0 TO 279
90  HPLOT I,I/2: NEXT I
100  VTAB 21: INPUT "PRESS RETURN TO END";A$
110  TEXT: END
```



Lo-Res



Hi-Res

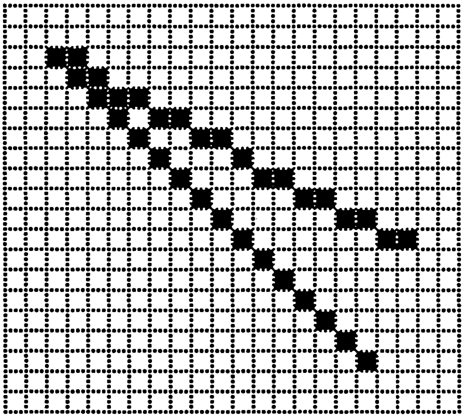
What's aliasing?

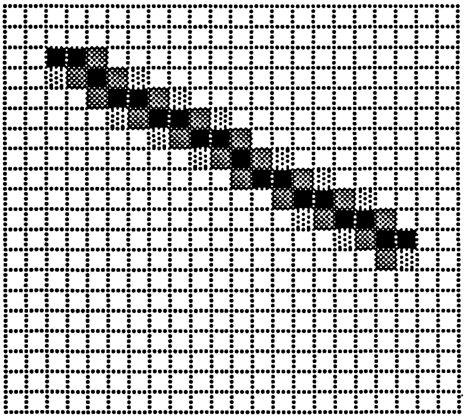
Aliasing refers to the jagged or stairstepping appearance of some lines.

Because the computer graphic screen is a grid of dots with relatively low resolution, smooth lines are not always possible. Only horizontal, vertical, and 45° diagonal lines can be totally smooth. All other lines must exhibit some degree of jaggedness or aliasing.

It's easy to demonstrate this phenomenon. Try to draw a 30° line across grid paper by connecting only the grid intersections. You'll get a jagged line. The lines will be smoother if they're drawn on grid paper with ten squares to the inch than they would be if drawn on paper with four squares to the inch. In the same way, lines drawn on the hi-res screens are smoother than lines on the lo-res screen—the greater the resolution, the less obvious the aliasing.

Sophisticated computer graphics systems feature anti-aliasing algorithms to reduce the jaggies. The most common algorithmn draws a thick line with varying intensities of color—only the center of the line is drawn in full intensity. This technique softens the edges of the line and eliminates the jagged appearance.





Should I use lo-res or hi-res graphics?

Most computer artists work with Apple's hi-res graphics rather than lo-res graphics.

Are there any advantages to low resolution graphics? For one thing, lo-res graphics can be used on any Apple, no matter how much memory it has. Also, a lo-res graphic uses only six sectors compared to the 33 sectors required for one hi-res graphic. Until recently, lo-res's claim to fame was its palette of 16 colors, but now many Apple users have access to double-hi-res which offers the same 16 colors.

Why do artists choose to work in hi-res? Resolution. The 40 × 48 resolution of lo-res doesn't allow fine line, detailed graphics. Double-lo-res offers 80 × 48 resolution, but 80 is still a far cry from 280, and even further from double-hi-res's 560! Higher resolution allows for more sophisticated images. Fortunately, standard hi-res colors can be dithered, or "mixed", so a creative artist can make a hi-res image look as if it contained the full spectrum of color.

Your choice of media, as with traditional fine arts tools, should be governed by what you're trying to achieve. You can create wonderful kaleidoscopic images in lo- and double-lo-res that will keep a viewer spellbound. Lo-res is also good for simple animated effects. The large-block, colorful images and letters are appealing to young children so there's a place in educational software for low-resolution. Also, graphics for the visually impaired might benefit from using the larger pixels.

For title screens, computer adventure game illustrations, anything that requires detailed representation, hi-res and double-hi-res are the choice.

Be aware too, that there are few commercial low-resolution graphics utilities. If you want to work in lo-res you'll probably have to do it with Applesoft BASIC commands.

Lo-res, hi-res, or double-res, it's up to the artist to use the tools creatively. We've seen one artist animate the standard lo-res screen so effectively that viewers were captivated as a swimmer (represented by a block of three pixels) encountered numerous perils as he dove from a diving board into a blue pool of pixels. The computer can be a marvelous artistic tool but it's you, the artist, who must make it so.

How do I see what's on the different graphics pages?

The commands GR, HGR and HGR2 take you to the lo-res graphics page, hi-res graphics page 1 and hi-res graphics page 2, respectively. The problem with these commands is that when they display these pages, or screens, they also clear them.

If you want to display these screens so that you can create a graphic, these commands are fine. If, however, you've BLOADed a graphic onto one of these pages and you want to see it, or edit it, you need to display the page without clearing it. To do this you'll have to POKE your way there.

Each POKE below acts as a switch to take you back and forth between the graphics pages without clearing them.

```
 POKE -16304,0 Text to Graphics
 POKE -16297,0 Lo-res to Hi-res
*POKE -16299,0 Page 1 to Page 2
 POKE -16300,0 Page 2 to Page 1
 POKE -16298,0 Hi-res graphics to Lo-res graphics
 POKE -16301,0 Graphics to Text
```

Depending upon where you are, and where you want to go, you can use a combination of POKEs, each separated by a colon. For example:

To go from text to the lo-res graphics page, type:
```
POKE - 16304,0
```
To go from text to hi-res graphics page 1, type:
```
POKE -16304,0: POKE-16297,0
```
To go from text to hi-res graphics page 2, type:
```
POKE -16304,0: POKE -16297,0: POKE -16299,0
```

Note: Poke -16304,0 is the graphics display switch. It'll put you into lo-res graphics when you first start up the Apple. Once you've used the HGR, or HGR2 command, at any point after booting, POKE -16304,0 will default to hi-res graphics, page 1, or page 2, whichever was displayed last.

*——> On the Apple //c or //e, in 80-column mode, POKE -16299,0 doesn't work. ESCAPE CONTROL Q will put you in normal 40-column mode and then you can use POKE -16299,0.

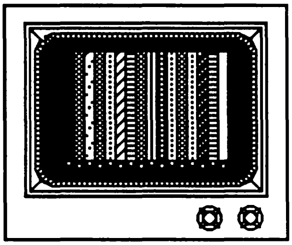
Why do I see the wrong colors on my CRT?

One of the problems with American (NTSC) transmission signals is that color information is relative not absolute. At one time or another, we've all turned on our television sets to find sickly green faces staring out at us. A simple twist of the color or hue adjustment cures this illness. It's no different with computer graphics.

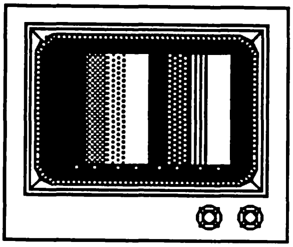
The color on television sets and composite color monitors must be adjusted to get the correct colors on the screen. Run the programs below to draw lo-res and hi-res color bars: adjust your screen image until the colors are accurate.

If you have an Apple //e or Apple //c, the lo-res program can be used to adjust the color for both double-lo-res and double-hi-res also, as the colors in both modes are identical to regular lo-res.

```
10 S=4
20 HOME:GR
30 FOR C=0 TO 15: COLOR=C
40 FOR X=0 TO 1
50 YLIN 0,39 AT X+S
60 NEXT X
70 HTAB S+1:VTAB22+T:PRINTC;
80 T=C/2=INT(C/2)
90 S=S+2
100 NEXT C
```



```
10 S=0
20 HOME:HGR
30 FOR C=0 TO 7: HCOLOR=C
40 FOR X=0 TO 34
50 HPLOT X+S,0 TO X+S,160
60 NEXT X
70 HTAB ((C+1)*5)-2: VTAB 22:PRINT C;
90 S=S+35
100 NEXT C
```



How do I use low-resolution graphics?

The low-resolution graphics page is made up of 40 horizontal X 40 vertical pixels. If you use the lo-res graphics page without the text window, the vertical resolution becomes 48. (See What's resolution? p. 122) Although lo-res can't be used to create finely detailed graphic images, there's a charm to the "block" look, and the palette of 16 vibrant colors beckons the Apple user to experiment with the graphics.

To display the lo-res page, type GR. The screen will default to a 40 × 40 black display, with a four-line text window at the bottom. You can turn on any one of the 1600 pixels by entering its X,Y coordinates with the command, PLOT X,Y. You tell Apple what color to use with the command COLOR=n, where n is the number of the color. The colors available are:

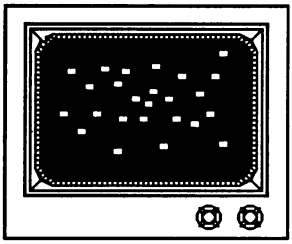
0 BLACK
1 MAGENTA
2 DARK BLUE
3 PURPLE
4 DARK GREEN
5 GREY
6 MEDIUM BLUE
7 LIGHT BLUE
8 BROWN
9 ORANGE
10 GREY
11 PINK
12 GREEN
13 YELLOW
14 AQUA
15 WHITE

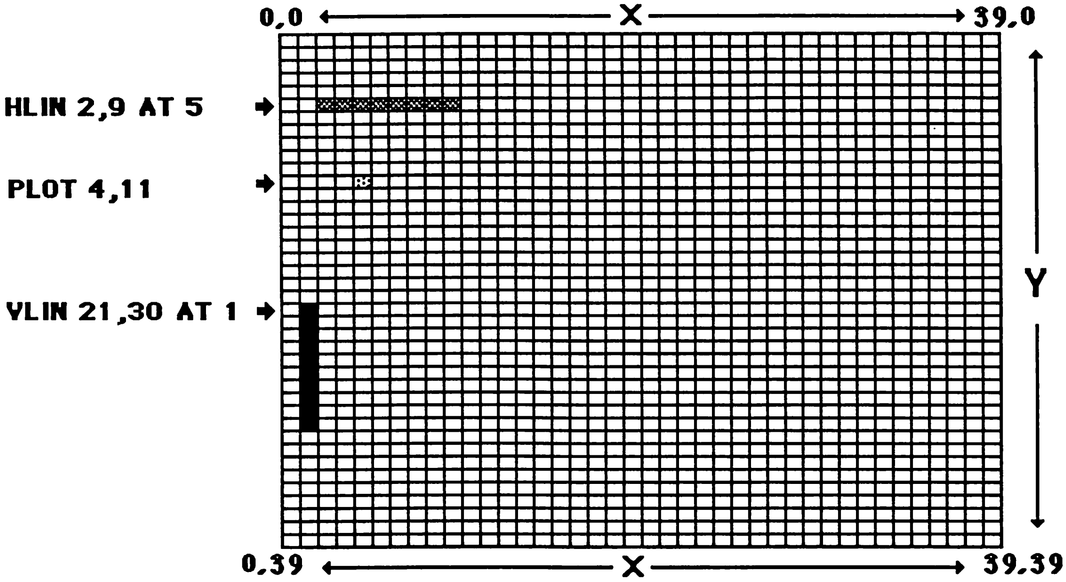
To draw a horizontal line on the lo-res page use the command, HLIN X1,X2 at n, where X1 is the beginning of the line, X2 is the end of the line, and n is the column it's to be drawn in.

Similarly, to draw a vertical line on the lo-res page, use the command, VLIN Y1,Y2 at n.

Here's a program that will plot pixels, randomly, on the lo-res screen.

```
10 GR
20 FOR I=1 TO 500
30 COLOR=INT(RND(1)*16)
40 X=INT(RND(1)*40)
50 Y=INT(RND(1)*40)
60 PLOT X,Y
70 NEXT I
```

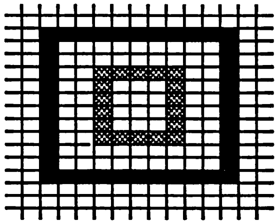
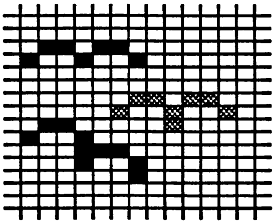
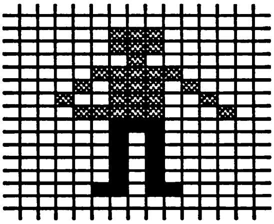




The best way to work with low-resolution graphics is to design the image on a piece of grid paper. Be sure to take into consideration the fact that the spaces on commercial grid paper are square, while the pixels on the lo-res screen are rectangular.

Here are some grid designs. Plot them on the Apple. Remember to use the color command first, or you'll find yourself plotting black (Apple's default color) on black, and wondering why you can't see anything.

Grid designs

How can I clear the entire lo-res screen to black?

The GR statement takes you to the low-resolution screen with a four-line text window at the bottom. Sometimes you may want to use the whole lo-res screen without the text window.

The full screen POKE (POKE -16302,0) doesn't quite work—it fills the bottom of the screen with grey lines rather than a black background. This is because a "space" character translates in lo-res to a black pixel on top of a grey pixel: to produce two black pixels you must print an inverse "@". (See Why do I get colors on the lo-res screen when I type? p. 130) A simple CALL will clear the screen to all black.

```
10 POKE -16302,0:POKE -16300,0
20 POKE -16298,0:POKE -16304,0
30 CALL -1998
40 GOTO 40
```

To stop this program, press RESET or CONTROL C.

There's no CALL to clear the lo-res screen to a color other than black.

How can I border the low-resolution screen?

Plotting borders on the lo-res screen is simple. Select any one of the 16 available colors and use the HLIN and VLIN commands.

Here's a program to plot a border in magenta:

```
10 GR : POKE -16302,0: CALL-1998: COLOR = 1
20 HLIN 0,39 AT 0:VLIN 0,47 AT 0
30 HLIN 0,39 AT 47:VLIN 0,47 AT 39
```

The horizontal and vertical (X,Y) coordinates can be changed to create different sized rectangular frames.

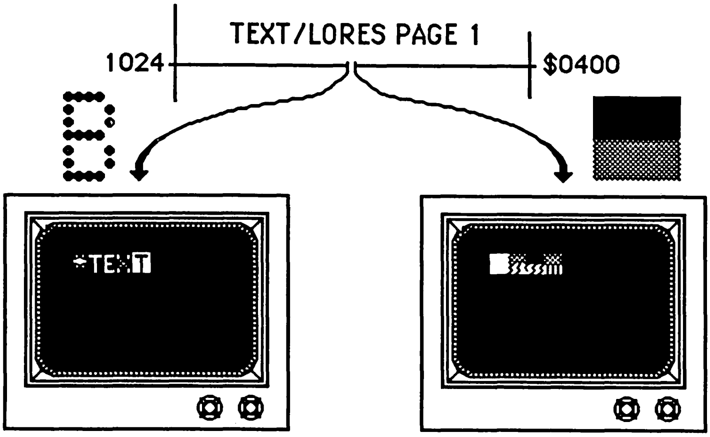
Why do I get colors on the lo-res screen when I type?

The low-resolution graphics page and the text page use the same block of Apple's memory—location 1024 ($0400). (See Memory map, p. 40) So, when you look at the lo-res screen you're seeing what you would see if you were looking at the text screen, but now, each byte is displayed as two blocks of color stacked one on top of the other.

Two stacked blocks of color are equivalent to one text character. While the resolution of the text screen is 40 × 24, the resolution of the lo-res screen is 40 x 48 . The difference in vertical resolution is because there are two blocks of color (vertically) to one character.

Each block can be any one of 16 colors. Every ASCII character has a different "pair" of colored blocks assigned to it. Thus, if you're looking at the lo-res display, and you type "B", you'll see a block that's dark blue on top and light green on the bottom. Typing "8" will display a solid pink block. Only 16 characters will display a solid color block (top and bottom colors the same)—all the other characters will display two-color blocks.

On the facing page is a lo-res color block chart. It tells you the colors of the stacked blocks for each upper-case ASCII character, in normal, flash and inverse modes. Refer to this chart if you want to use text commands to create lo-res graphics. (See Can the text page be used for animation? p. 180)



Low-resolution color blocks

A= ASCII character T=number of top color B=number of bottom color

A	Inverse		Flash		Normal		A	Inverse		Flash		Normal	
	T	B	T	B	T	B		T	B	T	B	T	B
@	0	0	0	4	0	12	P	0	1	0	5	0	13
A	1	0	1	4	1	12	Q	1	1	1	5	1	13
B	2	0	2	4	2	12	R	2	1	2	5	2	13
C	3	0	3	4	3	12	S	3	1	3	5	3	13
D	4	0	4	4	4	12	T	4	1	4	5	4	13
E	5	0	5	4	5	12	U	5	1	5	5	5	13
F	6	0	6	4	6	12	V	6	1	6	5	6	13
G	7	0	7	4	7	12	W	7	1	7	5	7	13
H	8	0	8	4	8	12	X	8	1	8	5	8	13
I	9	0	9	4	9	12	Y	9	1	9	5	9	13
J	10	0	10	4	10	12	Z	10	1	10	5	10	13
K	11	0	11	4	11	12	[	11	1	11	5	11	13
L	12	0	12	4	12	12	\	12	1	12	5	12	13
M	13	0	13	4	13	12	]	13	1	13	5	13	13
N	14	0	14	4	14	12	^	14	1	14	5	14	13
O	15	0	15	4	15	12	_	15	1	15	5	15	13
	0	2	0	6	0	10	0	0	3	0	7	0	11
!	1	2	1	6	1	10	1	1	3	1	7	1	11
"	2	2	2	6	2	10	2	2	3	2	7	2	11
#	3	2	3	6	3	10	3	3	3	3	7	3	11
$	4	3	4	6	4	10	4	4	3	4	7	4	11
%	5	3	5	6	5	10	5	5	3	5	7	5	11
&	6	3	6	6	6	10	6	6	3	6	7	6	11
'	7	3	7	6	7	10	7	7	3	7	7	7	11
(	8	3	8	6	8	10	8	8	3	8	7	8	11
)	9	3	9	6	9	10	9	9	3	9	7	9	11
*	10	3	10	6	10	10	:	10	3	10	7	10	11
+	11	3	11	6	11	10	;	11	3	11	7	11	11
,	12	3	12	6	12	10	<	12	3	12	7	12	11
-	13	3	13	6	13	10	=	13	3	13	7	13	11
.	14	3	14	6	14	10	>	14	3	14	7	14	11
/	15	3	15	6	15	10	?	15	3	15	7	15	11

Note: Control characters and lowercase letters will produce other colors.

Graphic Bonus: Lo-Res Text

Because lo-res pixels are so large, you can't get many words on the screen at one time. The following program will put letters on the lo-res screen.

```
10 GR: POKE -16302,0: CALL -1998
30 X=1: Y=2: REM starting X,Y coordinates
40 SP=1: REM spacing between letters
50 NY=10: REM number of vertical dots in characters
60 READ NC: REM number of characters
62 DIM C(NC)
64 FOR I = 1 TO NC: READ C(I):NEXT I:REM read colors
70 FOR I = 1 TO NC
75 COLOR=C(I)
80 FOR J = 1 TO NY: READ A$
90 FOR K = 1 TO LEN(A$)
100 IF MID$(A$,K,1)="X"THEN PLOT X,Y
110 X=X+1
120 NEXT K
130 X=X-LEN(A$): Y=Y+1: IF MX<LEN(A$) THEN MX=LEN(A$)
140 NEXT J
150 X=X+SP+MX: Y=Y-NY:MX=0
160 NEXT I
990 DATA 4: REM number of characters to print
995 DATA 14, 12, 13, 11: REM color of each character
1000 DATA "..XXXXX"
1005 DATA ".X.....X"
1010 DATA "X.......X"
1015 DATA "X.......X"
1020 DATA "X.......X"
1025 DATA "XXXXXXXXX"
1030 DATA "X.......X"
1035 DATA "X.......X"
1040 DATA "X.......X"
1045 DATA "X.......X"
1060 DATA "XXXXXXXX"
1065 DATA "X.......X"
1070 DATA "X.......X"
1075 DATA "X.......X"
1080 DATA "XXXXXXXX"
1085 DATA "X.......X"
1090 DATA "X.......X"
```

```
1095 DATA "X.......X"
1100 DATA "X.......X"
1105 DATA "XXXXXXXX"
1120 DATA ".XXXXXXXX"
1125 DATA "X"
1130 DATA "X"
1135 DATA "X"
1140 DATA "X"
1145 DATA "X"
1150 DATA "X"
1155 DATA "X"
1160 DATA "X"
1165 DATA ".XXXXXXXX"
1170 DATA "XXXXXXXX."
1175 DATA "X.......X"
1180 DATA "X.......X"
1185 DATA "X.......X"
1190 DATA "X.......X"
1195 DATA "X.......X"
1200 DATA "X.......X"
1205 DATA "X.......X"
1210 DATA "X.......X"
1220 DATA "XXXXXXXX."
```

How can I use lo-res page 2?

The second low-resolution screen is located in the same memory locations that BASIC uses to store programs. (See What's a memory map? p. 40) To use lo-res page 2, then, you must first relocate your BASIC program higher in memory.

Here's a program that will change BASIC's start of program pointers:

```
10 LOC=24576
20 HA=LOC/256
30 LA=LOC-HA*256
40 POKE LOC-1,0: POKE 103,LA: POKE 104,HA
50 PRINT CHR$(4);"RUN LORES2"
```

Warning! Be sure to save the above program before running it! Once the start of program pointers are changed, the program will disappear and the program LORES2 will be loaded and executed. Save the next program as LORES2.

```
10 IF PEEK(104)>12 THEN GOTO 100
20 PRINT "YOU MUST RELOCATE THE"
30 PRINT "BASIC PROGRAM AREA BEFORE"
40 PRINT "RUNNING THIS PROGRAM."
50 STOP
100 POKE -16302,0: POKE -16299,0
110 POKE -16298,0: POKE -16304,0
120 FOR I = 2048 TO 3063
130 POKE I,0: NEXT
140 C=RND(1)*256
150 L=RND(1)*1016
160 POKE 2048+L,C
170 IF PEEK (-16384)>128 THEN 200
180 GOTO 140
200 TEXT: HOME
```

Press any key to stop this program.

The main problem with lo-res page 2 is that there's no plotting support—COLOR=, PLOT, VLIN, and HLIN commands do not work on page 2. If you want to use this page, you have two options: POKE the correct pixel data or create the graphic on lo-res page 1 and BLOAD it to page 2 when you want to use it.

Graphic Bonus: Kaleidoscope

The Applesoft lo-res commands can create interesting random graphics, including kaleidoscopes. Here's our version:

```
10 GR
20 X = INT(RND(1)*20)
30 Y = INT(RND(1)*20)
40 C = INT(RND(1)*16)
50 COLOR= C
60 PLOT X,Y:REM upper left
70 PLOT 39-X,Y:REM upper right
80 PLOT X,39-Y:REM lower left
90 PLOT 39-X,39-Y:REM lower right
100 IF PEEK(-16384)>128 THEN END:REM look for keypress
110 GOTO 20
```

To stop this program press any key (Line 100). The program generates a random number for X,Y, and C. X and Y are used to plot a single pixel in the four quadrants of the screen, and C becomes the new plotting color. The program then loops back and generates new coordinates and a new color.

Note that RND(1) produces a "random number" between 0 and .99999999 (don't hold us to the number of 9's). The algorithm that generates the variables multiplies the random number by *one more* than the maximum value we can use (19 for the coordinates and 15 for the colors). After multiplying, INT chops off any decimals that may have been generated and a whole number is assigned to X, Y, and C.

A few simple variations yield entirely different patterns. One variant is to limit the number of color changes by adding this line to the program: 35 IF RND(1)>.3 THEN 50. Now, a new color is generated only if RND(1) in line 35 is less than .3. Try changing this value. Another variant uses the VLIN and HLIN instructions to produce boxes. Change lines 60 through 90 to read:

```
60 HLIN X,39-X AT Y:REM UPPER HLIN
70 HLIN X,39-X AT 39-Y:REM LOWER HLIN
80 VLIN Y,39-Y AT X:REM UPPER VLIN
90 VLIN Y,39-Y AT 39-X:REM LOWER VLIN
```

If you have a double-res plotting package such as HGR6 or Beagle Graphics, you can use the double-lo-res screen to produce larger kaleidoscopes.

How do I use high-resolution graphics?

The Apple II computer offers two separate areas or pages (see What's a Page? p. 34) for high-resolution graphics. We refer to these areas as hi-res page 1 (HGR) and hi-res page 2 (HGR2).

High-resolution graphics page 1 is made up of 280 horizontal x 160 vertical pixels, with four lines of text at the bottom of the screen. If you eliminate the text window, the vertical resolution becomes 192. High-resolution graphics page 2 doesn't default to a text window at the bottom of the screen. To get a text window, you must relocate your Applesoft program and POKE in the text—BASIC print statements can't be used. The resolution of hi-res page 2 is 280 X 192.

Hi-res offers considerably greater resolution than lo-res, but the extra memory used for this resolution must be made up somewhere. The trade-off is in color—only four colors plus black and white are available.

To display hi-res page 1, type HGR. The screen will default to the 280 × 160 display, with a four-line text window at the bottom.

To display hi-res page 2, type HGR2. The screen will default to the 280 × 192 display.

You can turn on any one of the 44,800 hi-res pixels (or 53,760 pixels without the text window) by entering its X,Y coordinates with the command HPLOT X,Y. You tell Apple what color to use with the command HCOLOR=n, where n is the number of the color. The colors available are:

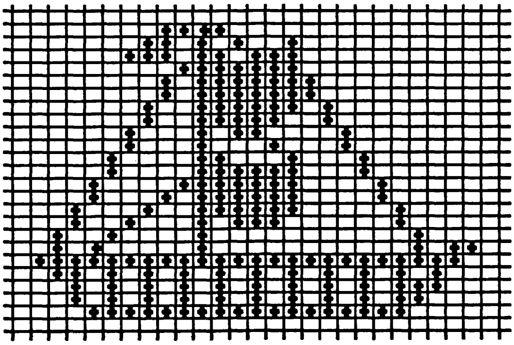
COLOR SET 1		COLOR SET 2	
0	BLACK	4	BLACK
1	GREEN	5	ORANGE
2	VIOLET	6	BLUE
3	WHITE	7	WHITE

To draw a horizontal, vertical, or diagonal line on the hi-res page use the command HPLOT X1,Y1 TO X2,Y2, where X1,Y1 is the beginning of the line and X2,Y2 is the end of the line. You can plot multiple connected lines with this command: HPLOT X1,Y1 TO X2,Y2 TO X3,Y3 TO X4,Y4.

While high-resolution graphics allow much finer lines and detail than low-resolution, we still get "jaggies" or aliasing. (See What's aliasing? p. 123)

Another problem with hi-res graphics involves the use of color. Colors can't be indiscriminately plotted next to each other without some surprising, and often disconcerting, results. (See How does the Apple produce high-resolution color? p. 138)

Here's a grid design. Plot it with the hi-res commands. Remember to use the HCOLOR command first, or you'll find yourself plotting black (Apple's default color) on black. The color of the boat will vary depending on whether you plotted on odd or even pixels. The sails will be white because odd and even pixels are plotted.



Note: If you intend to use the high-resolution graphics pages within BASIC programs be aware of the fact that Applesoft does not protect hi-res memory. Large programs with many variables can infringe upon hi-res memory and destroy the graphics display unless you protect it. (See What's HIMEM and LOMEM? p. 101, and How do I relocate programs? p. 103)

Try the following program:

```
10 HGR
20 FOR I= 1 TO 500
30 COLOR = INT (RND(1)*7)
40 X=INT(RND(1)*280)
50 Y=INT(RND(1)*160)
60 HPLOT X,Y
70 NEXT I
```

How does the Apple produce high-resolution color?

To understand how Apple produces colors, let's begin by seeing how other computers do it.

Most computers store color information in a pixel—each pixel has two properties: color and position. Position is determined by where the data representing the pixel is in graphics memory. Color is another story. To generate two-color graphics, one bit is used per pixel:

▯ **color 1**   ▮ **color 2**

If the bit is on ▯, color 1 is used; if the bit is off ▮ color 2 is used. For four-color graphics, two bits per pixel are needed:

▯▯ **color 1**   ▮▯ **color 3**

▯▮ **color 2**   ▮▮ **color 4**

Generating color this way requires memory—lots of it.

The Apple does things a little differently—it parlays position information into color.

In Apple's hi-res mode, one bit represents a pixel. If the pixel lies on an even X coordinate it's displayed in violet; if the pixel lies on an odd X coordinate it's displayed in green. Two adjacent pixels display white. In color, then, the actual resolution of the hi-res screen is 140 × 192, not 280 × 192, since every other pixel is used.

But what about the Apple's other hi-res colors?

For technical reasons, such as the speed of the Apple's CPU and how video must be produced, not all bits on the Apple's screen are shown. In each byte, one bit is unseen. The unseen bit determines if the *whole* byte is to contain violet and green pixels or blue and orange pixels. This unseen bit, often called the color bit, further restricts the resolution of the Apple: there are only 39 places across the screen where green and orange pixels, or violet and blue pixels can be adjacent to each other. (There are 40 bytes across the Apple hi-res screen.)

The following Applesoft program demonstrates the Apple's colors and color incompatabilities.

```
10 HGR: POKE -16302,0
20 HCOLOR=3: REM color is white 1
30 FOR X = 0 TO 138 STEP 2
40 HPLOT X,0 TO X,95: NEXT X
50 FOR X = 141 TO 279 STEP 2
60 HPLOT X,0 TO X,95: NEXT X
70 HCOLOR=7: REM color is white 2
80 FOR X=0 TO 138 STEP 2
90 HPLOT X,96 TO X,191: NEXT X
100 FOR X = 141 TO 279 STEP 2
110 HPLOT X,96 TO X,191: NEXT X
```

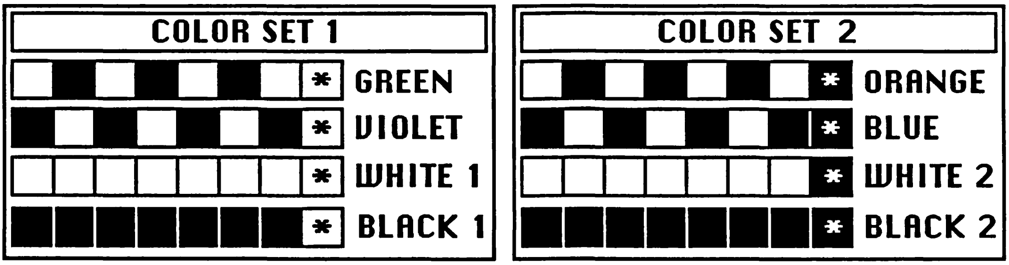
Note that the only colors being plotted are white 1 and white 2—the color is generated by the STEP statements. The loops in lines 30-40 and lines 80-90 plot only on even pixels; the loops in lines 50-60 and 100-110 plot only on odd pixels. Skipping pixels results in color even if Applesoft thinks we're plotting in white.

Note also that green and orange don't appear in the same screen positions even though they're plotted in the same positions. This peculiarity has to do with the way video is produced and can be used to increase the Apple's resolution on a black-and-white monitor.

Add the following lines to the program to demonstrate the incompatibility of the color sets:

```
120 HCOLOR=1:HPLOT 0,0 TO 279,191
130 HCOLOR=5:HPLOT 0,191 TO 279,0
```

The right side of the screen will show the incompatablity of green (HCOLOR 1) and orange (HCOLOR 5).



Why are there two whites and two blacks in hi-res?

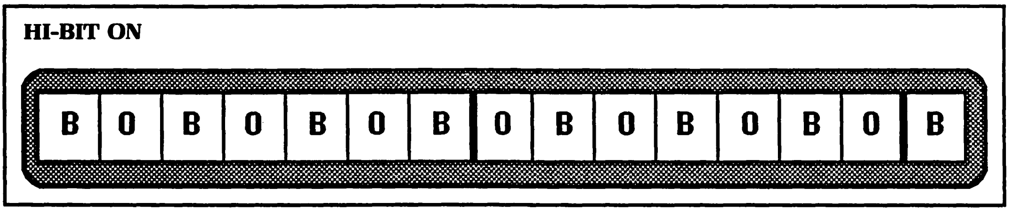
Apple has two sets of four colors.

Each bit is represented as a pixel on the hi-res screen. If a bit is turned on, a pixel is lit. If every pixel is lit you see white. If every pixel is turned off, you see black. If every other pixel is turned on, you see color.

This scheme would lead one to believe that there can only be one white and one black. However, the bits are located in the bytes of screen memory and in each byte, one bit isn't displayed. This bit determines whether the pixels in the byte will be in color set 1 (green or violet) or in color set 2 (orange or blue). This bit is called the color bit, but is more properly called the hi-bit because it's the seventh bit in the byte. If it's a 0, color set 1 is shown and the byte's numeric value will be less than 128. If it's a 1, color set 2 is shown and the byte's numeric value is greater than 127.

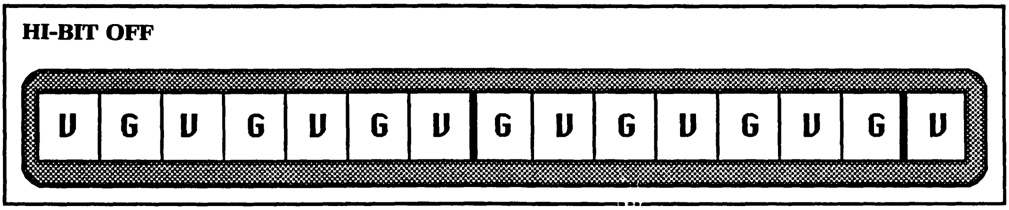
Plotting any color from color set 2, including white and black, changes all the pixels in the byte to color set 2. Plotting any color from color set 1, including white and black, changes all pixels in the byte to color set 1. The problem then, is not with the whites or blacks—they're identical to each other—it's with the colors of the pixels that surround them.

The diagram below represents two bytes and one bit at the upper left-hand corner of the screen (coordinate 0,0).



HI-BIT ON

B	O	B	O	B	O	B	O	B	O	B	O	B	O	B



HI-BIT OFF

V	G	V	G	V	G	V	G	V	G	V	G	V	G	V

How can I clear the hi-res screen to a color?

The HGR and HGR2 commands clear the screen to black. To clear the screen to another color, you must use a subroutine that's buried in the BASIC ROM.

Before the subroutine can be used, you must set the color with HCOLOR= and one HPLOT statement. Once set, a CALL will clear the screen to the desired color. Try this:

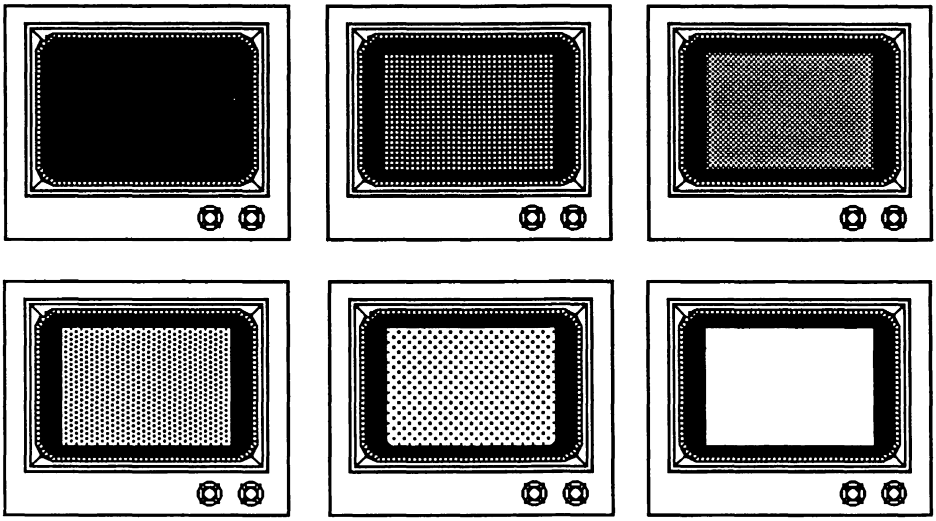
```
10 HCOLOR=1: HPLOT 0,0
20 CALL 62454
30 POKE -16302,0: POKE -16300,0
40 POKE -16297,0: POKE -16304,0
```

If you want to clear hi-res page 2, insert:

```
5 POKE 230,64
```

and change line 30 to read

```
30 POKE -16302,0: POKE -16299,0.
```



How can I put text on the hi-res screen?

The Apple's character generator can put text only on the text screen. If you want text combined with your "painted" graphics, use a paint program that has labeling capabilities, or specialized hi-res character software.

Your programmed graphics can produce text if you use a character generator that reroutes the text output from your program onto the hi-res screen. These fall into two broad groupings: block character generators and shape character generators.

Block generators produce characters in predefined blocks—usually 7 X 8 for small characters and 14 X 16 for large characters. They have features much like Apple's standard text commands, such as VTAB and HTAB positioning. Shape character generators use Applesoft shape tables to produce their characters so a character need only be as wide as its design requires. These character generators tend to print slower than block generators and they may not use standard Applesoft commands to print, but they usually feature the option to rotate and scale the characters.

If you have only a small amount of text to put on the hi-res screen you can use the graphic bonus program Hi-Res Text; thus you can have your text and your graphics too.

The data statements in lines 1000-1460 define the characters. You can change the data statements to create any letters or symbols you want. (Each one must be the same height as indicated in the variable NY in line 40).

This program shows only one solution for displaying text on the hi-res screen. If you have lots of text to put on the screen or you want to use fancy fonts, use one of the commercial character generators. (See About the artist's tools, p. 207)

Graphic Bonus: Hi-Res Text

```
10 HGR:POKE -16302,0: HCOLOR=3
20 X=10:Y=20:REM starting X,Y coordinates
30 SP=97:REM spacing between letters
40 NY=11:REM number of vertical dots in characters
50 READ NC:REM number of characters
60 FOR I = 1 TO NC
70 FOR J = 1 TO NY:READ A$
80 FOR K = 1 TO LEN(A$)
90 IF MID$(A$,K,1)="X"THEN HPLOT X,Y
100 X=X+1
110 NEXT K
120 X=X-LEN(A$):Y=Y+1:IF MX<LEN(A$) THEN MX=LEN(A$)
130 NEXT J
140 X=X+SP+MX:Y=Y-NY:MX=0
150 NEXT I
990 DATA 2:REM number of characters to print
1000 DATA "...XXX"
1005 DATA "..X...X"
1010 DATA ".X.....X"
1015 DATA "X.......X"
1020 DATA "X.......X"
1025 DATA "X.......X"
1030 DATA "XXXXXXXXX"
1035 DATA "X.......X"
1040 DATA "X.......X"
1045 DATA "X.......X"
1050 DATA "X.......X"
1060 DATA "XXXXXXXX"
1065 DATA "X.......X"
1070 DATA "X.......X"
1075 DATA "X.......X"
1080 DATA "X.......X"
1085 DATA "XXXXXXXX"
1090 DATA "X.......X"
1095 DATA "X.......X"
1100 DATA "X.......X"
1105 DATA "X.......X"
1110 DATA "XXXXXXXX"
```

How can I border the hi-res screen?

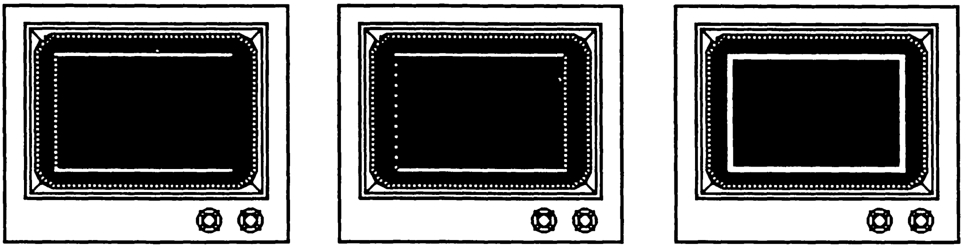
Plotting borders on the high-resolution screen should simply be a matter of using the correct commands to draw horizontal and vertical lines. Try the following program to create a white border:

```
10 HGR: POKE -16302,0: HCOLOR = 3
20 HPLOT 0,0 TO 279,0 TO 279,191 TO 0,191 TO 0,0
```

You do get a border, but the border you get is tricolored. The horizontal is white, as specified in the HCOLOR statement, but the vertical lines vary—violet on the left and green on the right. Why? Remember how the Apple produces color. (See How does the Apple produce high-resolution color? p. 138) A single column of pixels always displays in color.

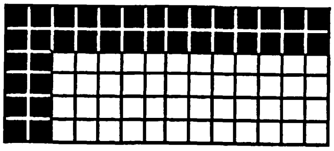
0 is an even number; 279 is an odd number. Since single pixels are either green or violet (or orange or blue) depending on whether they're plotted on even or odd pixels, the border is inconsistent.

To make clean white borders, plot double lines. Colored borders also look better plotted double, but they can be one pixel wide as long as you choose the correct even or odd pixels for the vertical lines.



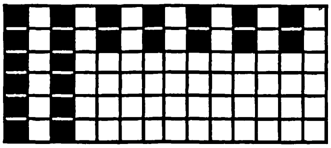
Graphic Bonus: Bordering the Hi-res Screen

White Border



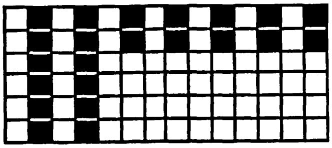
```
10 HGR: POKE -16302,0: HCOLOR = 3
20 HPLOT 0,0 TO 279,0 TO 279,191 TO 0,191 TO 0,0
30 HPLOT TO 1,1 TO 278,1 TO 278,190 TO 1,190 TO 1,1
```

Violet or Blue Border



```
10 HGR: POKE -16302,0: HCOLOR = 2
20 HPLOT 0,0 TO 278,0 TO 278,191 TO 0,191 TO 0,0
30 HPLOT TO 2,1 TO 276,1 TO 276,190 TO 1,190 TO 2,1
```

Green or Orange Border



```
10 HGR: POKE -16302,0: HCOLOR = 1
20 HPLOT 1,0 TO 279,0 TO 279,191 TO 1,191 TO 1,0
30 HPLOT TO 3,1 TO 277,1 TO 277,190 TO 3,191 TO 3,1
```

To make borders of different sizes, be sure to plot white on adjacent pixels, green or orange on odd pixels only, and violet or blue on even pixels only.

How can I get the Apple to draw circles?

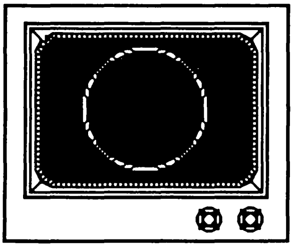
Unfortunately, Applesoft doesn't provide the CIRCLE instruction that is present in many other BASICS. All is not lost: you can draw a circle by using the correct trigonometric functions. If you're the type who panics at the thought of trig, don't. You can easily use the program below.

AR stores the aspect ratio, so if the circle doesn't look quite circular to you, try changing AR to .9 or .8. Be aware that every CRT model will display a differently proportioned circle, so the correct aspect ratio for you will depend upon what monitor or television you're using.

PI is...well..., PI.

XC and YC are the X and Y centers of the circle. XR and YR are the X and Y radius of the circle. Change these to move the circle or alter its size. If you make XR bigger or smaller than YR, you'll get ovals!

The loop steps through the angles from 0 to 360 (actually it misses a few but these are absorbed in the resolution of the screen). Angles, in Applesoft, are measured in radians, and 0 to PI X 2 represents a full 360° rotation. If you change the step size, you'll get a circle made of dots.

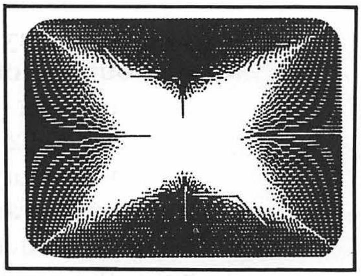


```
10 HGR: HCOLOR=3
20 AR=1: PI=3.1418
30 XC=140: YC=100
40 XR=50: YR=YR*AR
50 FOR Z=0 TO PI*2 STEP .02
60 X=XC+SIN(A)*XR
70 Y=YC+COS(A)*YR
80 IF X>279 OR X<0 OR Y>191 OR Y<0 THEN 100
90 HPLOT X,Y
100 NEXT A
```

Graphic Bonus: Moire

The Applesoft HPLOT command can be used to produce interesting graphic effects. Type in the following program:

```
10 HGR:POKE-16302,0
20 HCOLOR=3
30 FOR X = 0 TO 279 STEP 4
40 HPLOT X,0 TO 279-X,191
50 NEXT
60 FOR Y = 191 TO 0 STEP -3
70 HPLOT 0,Y TO 279,191-Y
80 NEXT Y
```



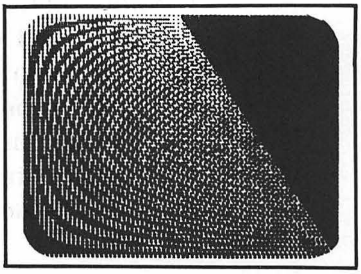
Although it may not appear so, this program plots straight lines in white. The colors are due to artifacting; the swirls are due to aliasing. The overall effect is termed a moire pattern.

Try the program with different values in the STEP instruction for different effects (line 30). If you want the artifacted colors to be orange and blue, set HCOLOR equal to 7 in line 20.

Moires can be effectively used for title screens. If you create a pattern you like, BSAVE the screen. The screen image can then be loaded into a paint program for enhancement.

Nonsymmetrical moires are also interesting. Try the following program. Be sure to erase the previous one from memory by using the command NEW. (Don't forget to SAVE it first.)

```
10 HGR:POKE -16302,0
20 HCOLOR=3
30 FOR X = 0 TO 279 STEP 4
40 HPLOT X/2,0 TO X,191
50 NEXT X
60 FOR X = 279 TO 0 STEP -4
70 HPLOT X,191 TO X,0
80 NEXT
```



To change the green background to violet, change line 60 to FOR X = 278 TO 0 STEP -4.

If you have a double-res plotting package such as HGR6 or Beagle Graphics, try these programs in double-hi-res.

What's double-resolution graphics?

The story of double-res starts in text mode.

The method of producing 80-column text with the Apple 80-column card for Apple //e computers is bizarre. More text on the screen uses more memory in the computer. To avoid radical changes to the memory map, Apple located the extra memory for 80-column text outside of the normal 64K 6502 address space. A little extra programming in the 80-column card ROM fooled the CPU.

The 6502 thinks it's storing pairs of characters in identical screen memory locations but the 80-column card circuitry routes the first character to aux memory on the 80-column card and the second character to the regular main memory. Video circuitry combines the main and aux memory to produce an 80-column display. This is why pressing CONTROL RESET when in 80-column mode jumbles the display ("?SYNTAX ERROR" changes to "SNA RO")—every other letter is in aux memory.

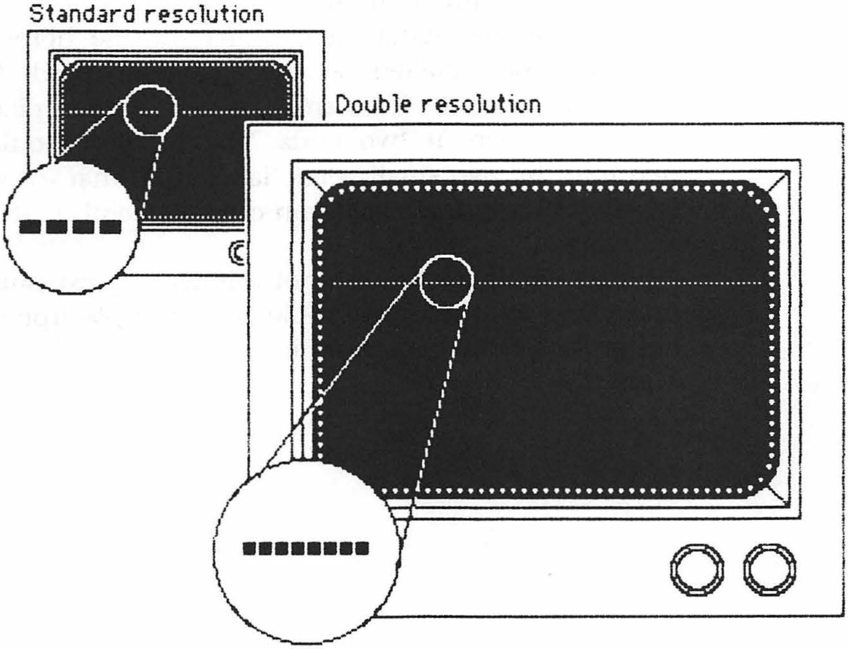
Lo-res graphics, which uses the same area of memory as text, benefits from combined main and aux memory too. The horizontal resolution is doubled—double-lo-res provides us with a resolution of 80 × 40 with the text window, and 80 × 48 without the text window.

Someone at Apple discovered that the video circuitry could be coaxed into doing the same thing with the hi-res graphic screen. Each byte on the regular hi-res graphic screen has a companion byte on the extended 80-column card (or its equivalent from another manufacturer). The structure of the screen is identical to that of the regular hi-res screen except for the shuffling of main and aux memory bytes. Double-hi-res graphics coordinate 0,0 is in aux memory, and coordinate 8,0 is in main memory. A double-hi-res drawing program must translate the coordinates it's receiving from the joystick or other input device into an address, and then store the correct bit in either main or aux memory to produce a pixel. With double-hi-res, the horizontal resolution of the hi-res screen is doubled to 560.

BASIC can't plot outside of the main memory area, so double-res isn't available from standard Applesoft. Further, because double-hi-res was more an accident than planned creation, when double-hi-res page 2 is activated, the entire main bank of memory is turned off. This plays havoc with programs—common techniques such as page flipping are not available except to very sophisticated programmers.

You can turn on the double-res display from BASIC and view the screen organization. Enter 80-column mode by keying PR#3 or ESCAPE 8. Type POKE -16290,0 to turn on double-resolution. Then type GR or HGR to go to the lo-res or the hi-res page. POKE -16291,0 turns off double-res.

Remember—Applesoft graphics commands will not work properly in double-resolution; they'll affect only main memory. (See How do I use double-resolution graphics? p. 151)



What do I need to get double-resolution graphics?

In order to have the extended graphics capabilities of double-resolution, you must use a 128K Apple with an extended 80-column card. Apple //c computers have this circuitry built-in and ready to go—if you're working with a //c you don't have to read the rest of this page.

Apple //e users must purchase the extended 80-column card as an option. The Apple //e must also have a "rev.B" main circuit board. You can identify the circuit board by turning the power off, removing the cover of the Apple //e, and looking at the lettering etched at the rear of the board behind the slot connectors. There you'll find a three-part serial number, something like 820-0064-A. The letter indicates the version of the circuit board. If it's a "B", you can go on to the next paragraph. If its an "A" and the computer has an extended 80-column card, an Apple dealer will provide a free exchange of the main circuit board to rev."B".

To enable the switching circuitry for the hi-res memory, a jumper must be installed on the 80-column card. A jumper is packed with the card—it's a small, square-shaped piece of plastic (about 2mm), open at two ends. The jumper should be gently pushed into the two small wires (labeled J1) that extend from the lower left corner of the 80-column card. Remember—turn the computer off first!

Unfortunately, at the time of this writing existing add-on 80-column cards for the older Apple II and Apple IIplus computers cannot provide double-res graphics.

How do I use double-resolution graphics?

If you're using an Apple //c or a 128K Apple //e, you can work with double-resolution graphics. Then all you need to draw or paint is a good double-res paint program. (See The artist's tools section, p. 207) Most of the commercial paint programs also include slide show utilities.

If you want more control over your double-res graphics than these utilities allow, you need an amplified Applesoft. Because Applesoft was created long before Apples had double-res graphics, normal Applesoft can't be used. There are, however, many commercial products that give you easy access to the double-res screens from BASIC.

There are several ways to add commands to Applesoft.

The most popular way to amplify Applesoft is through the "ampersand hook." The ampersand (&) is a valid Applesoft command. Most of the time nothing happens when you type an ampersand or use it in a program, but if the hook has been connected, something will happen—Applesoft will jump to a machine language subroutine and execute that subroutine.

The Applesoft RENUMBER utility on the DOS 3.3 System Master and APA on the DOS Toolkit are two programs that use the ampersand hook. Beagle Graphics and CAT Graphics offer ampersand routines that give you access to double-res. As a bonus, these packages also have other useful features such as block move routines.

Another approach to amplifying Applesoft is to actually modify Applesoft. The modified Applesoft is loaded into the language card or bank-switched memory of the Apple and is used just as you would use regular Applesoft. Programs that use this approach, such as HGR6 and Doublestuff, generally add a few commands that parallel the normal graphics commands. Beginners may find this type of program the best place to start.

How does the Apple produce double-hi-res color?

Creating color information in double-high-resolution is similar to the technique used in standard high-resolution. Four bits represent a pixel, and 16 different colors can be represented in four bits.

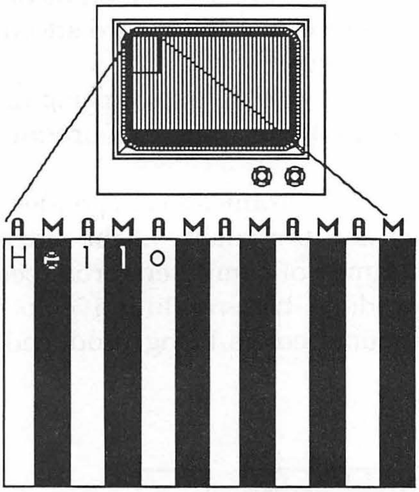
■■■■	**black**	□■■■	**brown**
■■■□	**red (magenta)**	□■■□	**orange**
■■□■	**dark blue**	□■□■	**grey 2**
■■□□	**purple**	□■□□	**pink**
■□■■	**dark green**	□□■■	**light green**
■□■□	**grey 1**	□□■□	**yellow**
■□□■	**medium blue**	□□□■	**aquamarine**
■□□□	**light blue**	□□□□	**white**

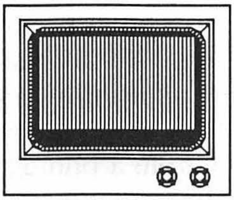
The screen position of the bit yields color. Thus, each color can appear in only 140 positions across the screen (560/4=140).

As in hi-res, one bit in each byte isn't displayed, but unlike hi-res, the unseen bit isn't used in double-hi-res, so you can put any color next to any other color without problems. The two greys are visually indistinguishable, but they may have different effects when mixing colors or when doing detailed work due to the different bit positions.

Anatomy of double-high-resolution

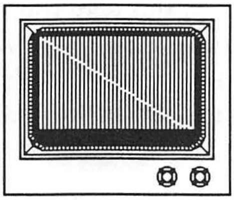
Display





```
PR#3
HGR
POKE -16290.0
```

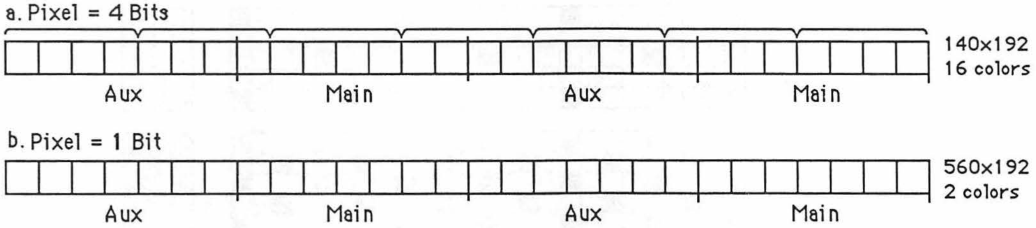
Only Main memory is cleared to black.



```
HCOLOR=3
HPLOT 0,0 TO 279,160
```

Applesoft only plots in Main memory.

Pixel Arrangement



Note: "Main" refers to the 64K on the Apple motherboard. "Aux" is the 64K on the extended 80-column card

Color Assignments

a. Pixel = 4 Bits

□□□□	0	■□□□	8
□□□■	1	■□□■	9
□□■□	2	■□■□	10
□□■■	3	■□□□	11
□■□□	4	■■□□	12
□■□■	5	■■□■	13
□■■□	6	■■■□	14
□■■■	7	■■■■	15

c. Pixel = 1 Bit

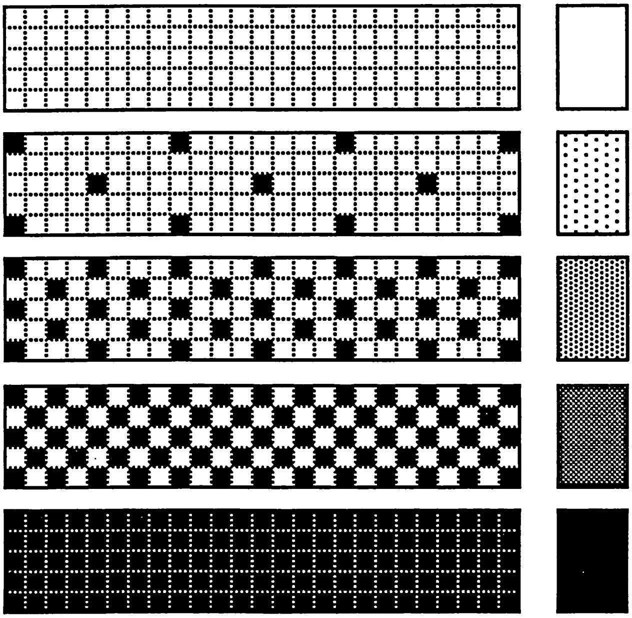
□	0
■	1

What's dithering?

Apple computer graphics are digital—pixels can either be on or off. With this limitation, then, all Apple graphics should be hard-edged, black and white images—not unlike woodcuts.

Dithering allows the Apple to display "greys" by arranging on and off pixels in tight patterns, similar to the technique of producing a halftone from a photograph.

The quality and range of the grey scale produced is dependent upon the resolution of the computer and the quality of the video display. Due to the way the Apple II family of computers produces color (See How does the Apple produce high-resolution color? p. 138), dithering may result in additional colors being produced. (See What's artifacting? p. 155)



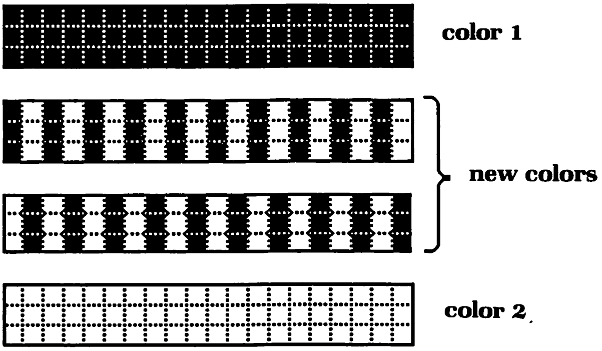
What's artifacting?

Artifacting is a computer graphics term that refers to the artificial production of color. Video engineers prefer the term "color anomalies."

In standard high-resolution, all color is the result of artifacting—pixels have no color properties. (See How does the Apple produce high-resolution color? p. 138) The colors are a result of how and when the video circuitry sweeps through the graphic page. Apple circuitry does not really produce a green, violet, blue, or orange signal.

Computers that have true color (like the Atari or IBM) can be coaxed into producing more colors by using artifacting. Plotting pixels on alternate screen positions may result in a new color as a result of the timing of the video signal.

On the Apple the term artifacting is sometimes applied to the creation of new colors by arranging the existing ones into a variety of patterns: alternating rows of green and orange, for example, to produce a yellowish hue. This is a crude and approximate application of the term. Most computer graphics artists describe the technique as "dithering." (See What's dithering? p. 154)

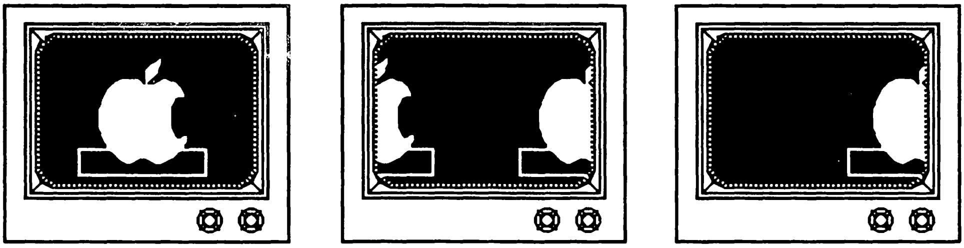


What's clipping and wrapping?

At the edges of the graphic screen—or the edges of the active graphics window—there are two options for programmers to take: wrapping or clipping.

Wrapping occurs when the graphic "turns over" and continues on the opposite side of the screen. This option is frequently used in games, animations, and other interactive applications. When you move Pacman to the edge of the screen, he wraps around the screen, reappearing at the opposite side, still pursuing his dots.

The other alternative is clipping. In clipping, the graphic stops at the edge. Clipping is usually preferred in paint programs and technical applications such as charting, graphing, and text, where the graphic must be carefully controlled.



wrapping

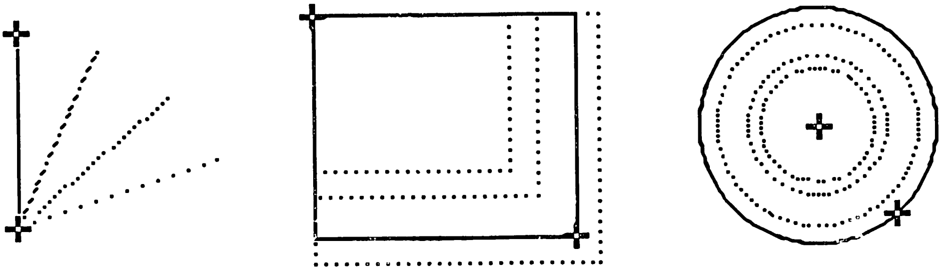
clipping

What's rubber banding?

When using the line option with a paint program, you usually select the beginning point and the end point of the line. It's difficult to guess if the line is too short, too long, angled too much, and so on. Rubber banding is a solution to this problem.

The rubber band is a temporary line. The line is displayed as you draw and move it with your input device, but it isn't permanently drawn until a button, key, or pen press is detected by the program. The angle and length of the line can be dynamically changed until it's exactly the way you want it.

Most software programs now use the rubber banding technique for circles, boxes, and other geometric figures as well as lines. It's a welcome improvement over guessing.



How do I design a font?

A font's main function is to be read. Fonts may be decorative, and through their "style" may impart a message, but a letter's flourishes should not detract from its legibility.

Fonts may be designed for a specific purpose, as was the font KEY that Michael designed for this book. It serves a double purpose: It lets you know when we're referring to keys on the keyboard, as in "press A," and it lets us refer to the special keyboard keys, for example, ESCAPE .

The key to successful font design is consistency. The curve of a "D" should reflect the curve of the "B". "W" is not an upside down "M"! If you're working from a type catalog or other source, observe the distinguishing characteristics of each font. (See Anatomy of letters, p. 159.)

If your font is to be displayed in color, be sure all upright strokes within the character are two pixels wide. (For double-hi-res, four pixels wide) The Apple's method of producing color by displaying every other pixel imposes this restriction; it can severely limit your font design.

To design a font, you'll need a font editor. Choose one that's compatible with the text entry program you'll be using. We're particularly fond of three editors: Higher Text for interactive, program-driven text generation, Fontrix for screen generation, and Apple Mechanic for shapetable fonts. Higher Text and Fontrix have their own character generators.

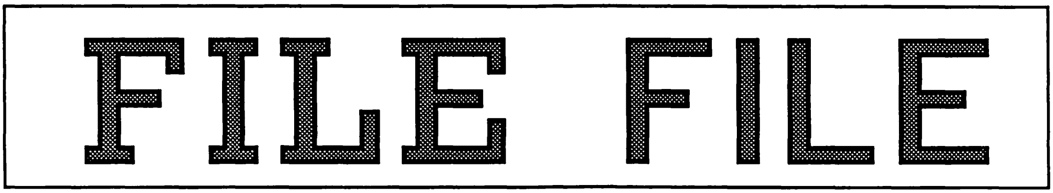
Higher Text is a nonproportional, fixed block-size generator. With it, you can design large (14 × 16) or small (7 × 8) characters that can be displayed in color. When designing large fonts with this program, allow three or four pixels for character descenders. Provide sufficient space between characters so that they don't bleed together—two or three pixels is usually adequate. (See Anatomy of a font, p. 160.) Because the fonts aren't proportional, kerning isn't possible—the "i" will swim within the 14 × 16 character block.

With Fontrix, a proportional character generator, there's a lot more freedom. Fonts can occupy blocks up to 32 × 32 pixels. The program's screen creation utility gives you control of spacing between characters and lines.

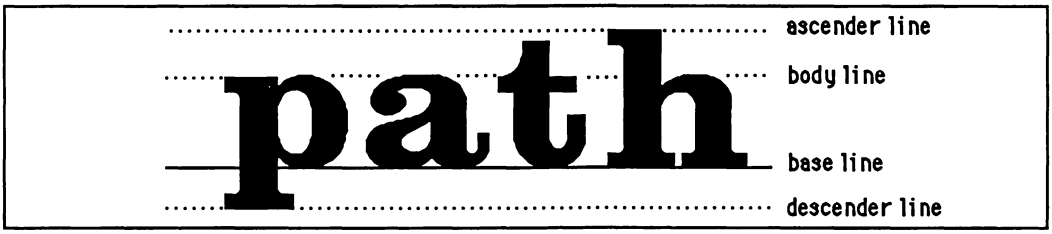
With a shapetable character generator, you theoretically have more freedom in font design than you would have in a block-style generator.

Anatomy of letters

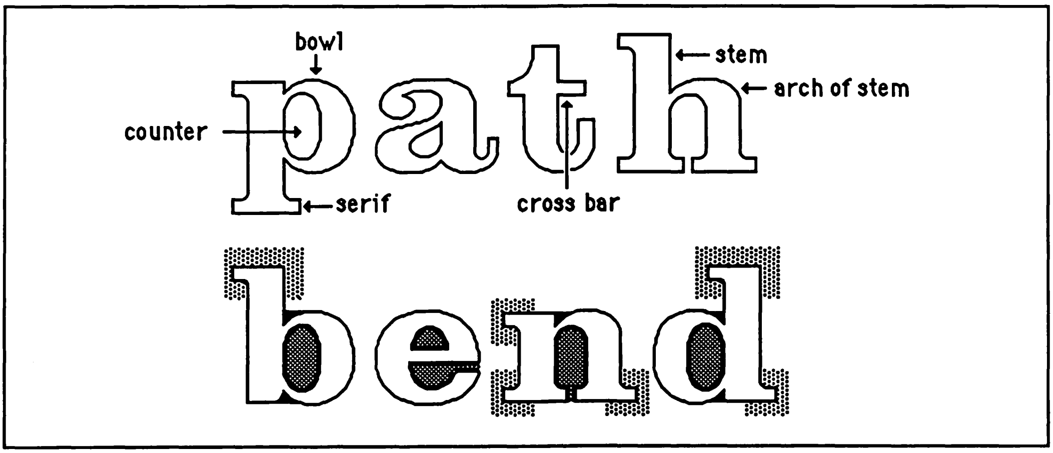
Serif and Sans serif fonts



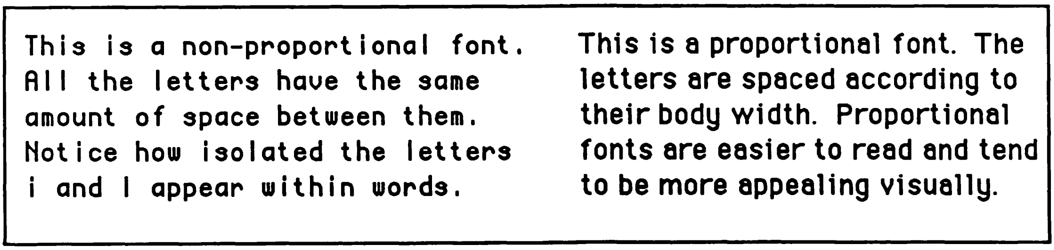
Lettering guidelines



Elements of letters



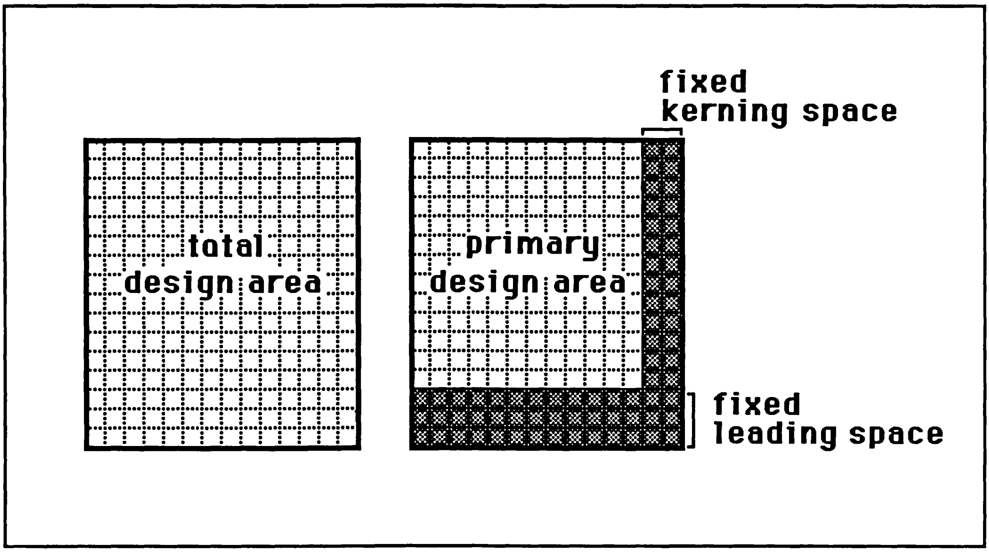
Proportional and non-proportional fonts

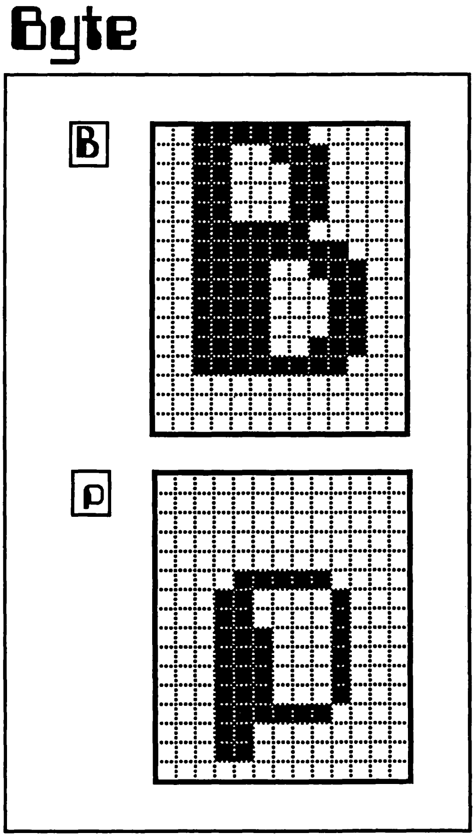


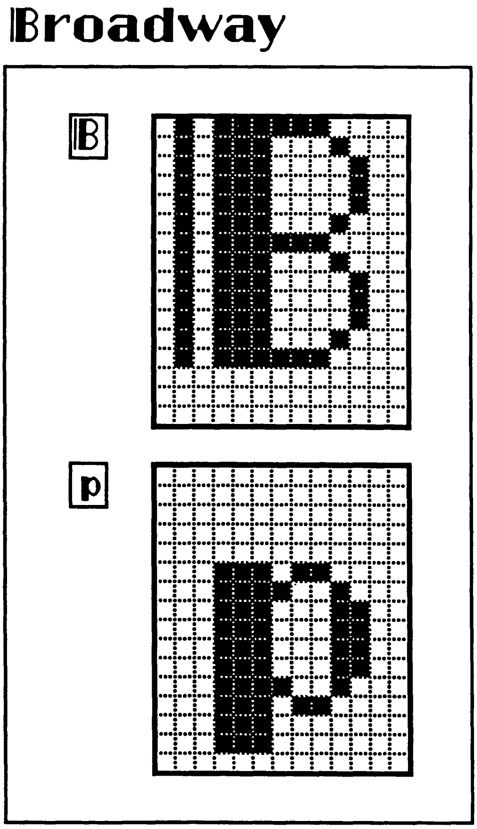
This is a non-proportional font. All the letters have the same amount of space between them. Notice how isolated the letters i and l appear within words.

This is a proportional font. The letters are spaced according to their body width. Proportional fonts are easier to read and tend to be more appealing visually.

Anatomy of a font







How many pictures will fit on a disk?

A formatted, initialized, single-sided DOS 3.3 disk contains 560 sectors, 496 of which are usable. The number of picture files that will fit on a disk depends upon a few factors. Assuming that there's nothing else on the disk:

Low-resolution pictures take up 6 sectors—you can save 496/6, or 82 pictures on one disk.

Double-low-resolution pictures take up 12 sectors—you can save 40 pictures on one disk.

High-resolution pictures may take up 33 or 34 sectors on a disk —you can save 14 pictures on a disk.

Double-high-resolution pictures use 68 sectors on a disk—you can save 7 pictures on a disk.

A formatted, initialized ProDOS disk contains 280 blocks, 221 of which are usable if the disk is bootable. On a data disk, which is nonbootable, 273 blocks are usable.

Low-resolution pictures take up 3 blocks—you can save 73 pictures on a bootable disk, or 91 pictures on a data disk.

Double-low-resolution pictures use 6 blocks—you can save 36 pictures on a bootable disk, or 45 pictures on a data disk.

High-resolution pictures use 17 blocks—you can save 13 pictures on a bootable disk, or 16 pictures on a data disk.

Double-high-resolution pictures take up 34 blocks—you can save 6 pictures on a bootable disk, or 8 pictures on a data disk.

You can fit more than the number of pictures indicated above, if the pictures are compressed or packed before they're saved. Packing a picture can save a considerable amount of disk space, but packed pictures can't be BLOADed onto the graphics screen—they must be unpacked for you to edit or look at them.

Many graphics software programs contain picture-packing (and unpacking) utilities. Depending upon the complexity of the image, a simple hi-res picture may pack from 34 sectors to less than 10 sectors of disk space.

How do I move pictures from disk to disk?

Pictures are saved to disk as binary files. You can't LOAD a picture into the computer's memory and then SAVE it to another disk as you would a BASIC program.

One way to transfer a picture file is to use a binary file copy utility, such as FID on the Apple DOS 3.3 System Master disk or FILER on the ProDOS User's disk.

If you don't have a binary file utility available, you can BLOAD the picture file from a disk into the computer's memory and then BSAVE it to another disk. When you BSAVE a binary file you must tell Apple both where the file is (its address in memory) and what the length of the file is. The procedure is as follows:

Put the disk with the picture file on it into the disk drive
Type: BLOAD name of picture,Address RETURN
Put the disk that you want to save the picture to into the drive
Type: BSAVE name of picture,Address,Length RETURN

The address and length of the file depends upon what graphics mode it was created in and what page it's loaded from. You can use either decimal or hexadecimal numbers.

Decimal	Hexadecimal
Low-resolution:	
BLOAD name,A1024	BLOAD name,A$400
BSAVE name,A1024,L1024	BSAVE name,A$400,L$400
High-resolution page 1:	
BLOAD name,A8192	BLOAD name,A$2000
BSAVE name,A8192,L8192	BSAVE name,A$2000,L$2000
High-resolution page 2:	
BLOAD name,A16384	BLOAD name,A$4000
BSAVE name,A16384,L8192	BSAVE name,A$4000,L$2000

Giving the length as 8192($2000) will BSAVE the hi-res picture as a 34 sector file. You can save it as a 33 sector file simply by changing the length of the file when you BSAVE it—instead of L8192 (L$2000), type L8187 (L$1FFB)

At present, there's no consistency in the way commercial software programs save these double-res images. It's best, therefore, to use a binary file utility to move double-res picture files from disk to disk.

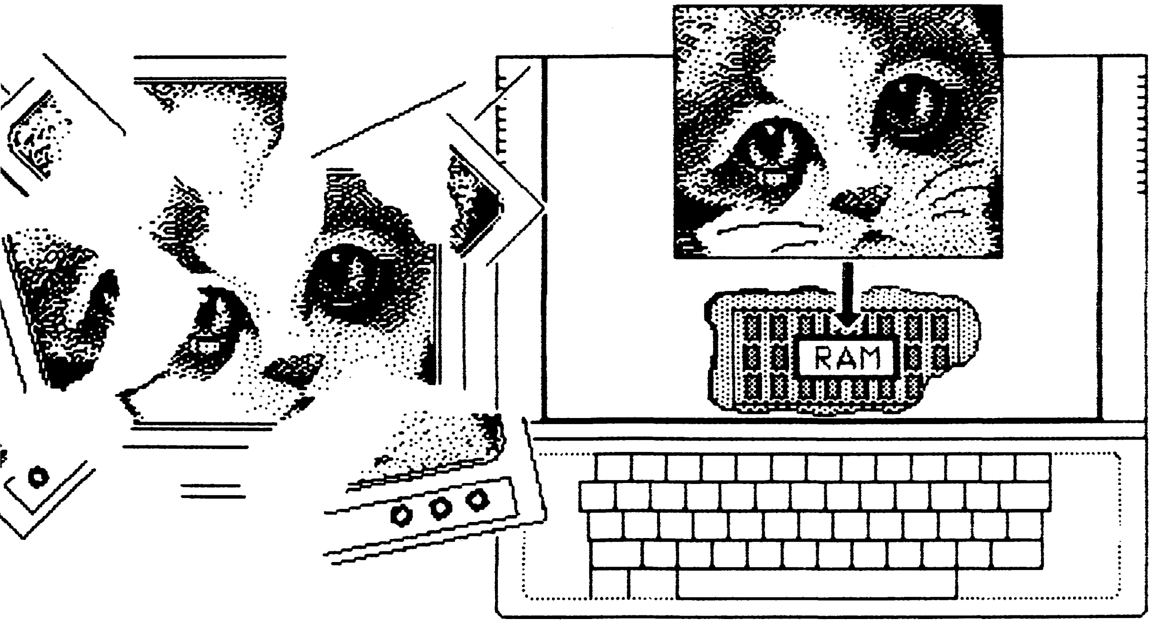
How can I save my pictures when the system crashes?

If a program crashes while you're working on a graphic, remove the program disk and insert a DOS 3.3 slave disk. Warm boot the disk (See What's the difference between a cold boot and a warm boot? p. 26.), POKE to the hi-res screen, and usually (unless the crash destroyed it) you'll find your graphic waiting. BSAVE it to disk by giving its name, address, and length (See How do I move pictures from disk to disk? p. 162), and reboot the graphics program you were using.

This procedure will work for shapes and shapetables too if you know where in memory you loaded them. This is a good reason for keeping records!

Warm booting ProDOS or DOS 3.3 master disks (the kind produced by the MASTER CREATE program on the DOS 3.3 System Master disk), instead of slave disks, will destroy hi-res memory.

Unfortunately, some commercial software, as part of their copy-protection schemes, don't allow you to quit their programs unless you cold-boot another disk. We try not to use programs that do this.



Why do pictures load like venetian blinds?

The Apple display screens aren't just blocks of memory; they have a definite and slightly bizarre structure. The way the screen is constructed is termed a screen map.

The Apple text screen actually consists of eight "super lines" of 128 characters or bytes. Watch the following program in action on a 40-column display:

```
10 HOME
20 FOR I = 1024 TO 2039
30 IF I/128<>INT(I/128) THEN GOTO 50
40 C=C+1:CH=ASC(CHR$(C))+192
50 POKE I,CH
60 NEXT I
```

Each of the eight lines is indicated by a letter. Note that the first line starts at the top of the screen, goes for 40 bytes, and then resumes one third of the way down the screen. If you count the A's, you'll find that there are only 120 of them—eight bytes at the end of each "super line" are not displayed. The unused bytes on the text page, however, are used. If you run this program with the Apple 80-column card active, you'll have to press RESET to regain control over the screen. Be careful when you POKE into this area.

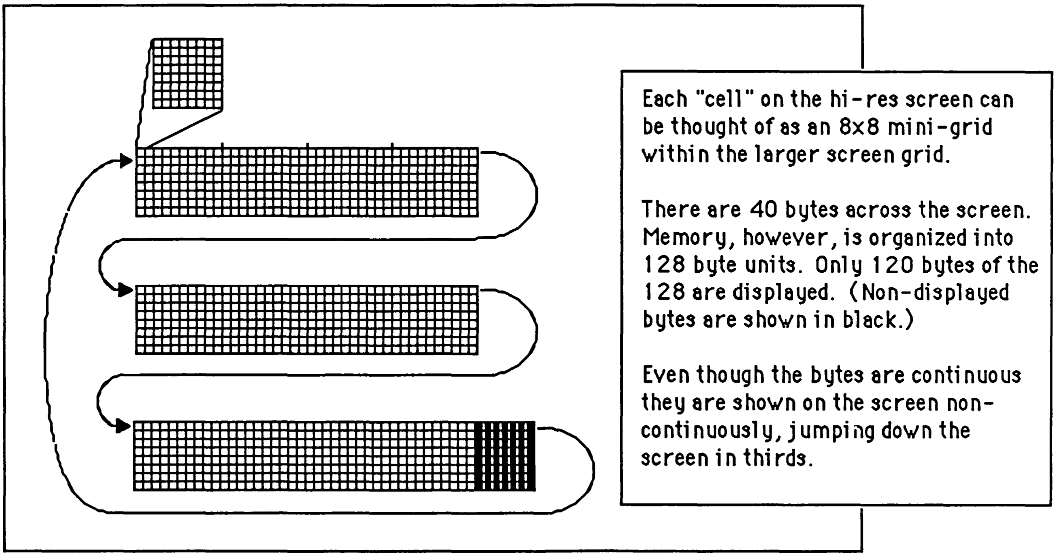
The structure of the hi-res screen is similar to, but more complicated than, that of the text screen. Instead of eight "super lines" there are 64. Here's the hi-res equivalent of the text screen program:

```
10 HGR:POKE -16302,0
20 FOR I = 8192 TO 16383
30 POKE I,255
40 FOR D = 1 TO 25:NEXT D
50 NEXT I
```

On the hi-res screen, unlike the text screen, each dot on the screen is addressable. On the text screen, an entire eight-dot-high character block was turned on—eight rows were sufficient to fill the screen. On the hi-res screen, the topmost pixels of the eight dots are turned on first, then the second row of pixels, and so on until the entire screen is white.

So, pictures load like venetian blinds because of the structure of the screen—rows of pixels aren't stored continuously in memory.

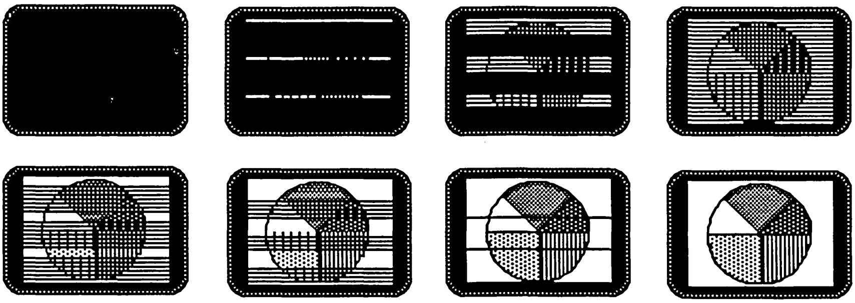
Anatomy of a hi-res screen



Each "cell" on the hi-res screen can be thought of as an 8x8 mini-grid within the larger screen grid.

There are 40 bytes across the screen. Memory, however, is organized into 128 byte units. Only 120 bytes of the 128 are displayed. (Non-displayed bytes are shown in black.)

Even though the bytes are continuous they are shown on the screen non-continuously, jumping down the screen in thirds.



Try this demonstration program and watch the order in which the screen turns white.

```
10 HGR: POKE -16302,0
20 FOR I = 0 TO 8192
30 POKE 8192+I,255
40 NEXT I
```

When a picture is BLOADed, it's read into memory contiguously, much as this program pokes continguous memory locations. Pictures load like venetian blinds!

Graphic Bonus: Slide Show

Here's a program all Apple artists can use. It creates a self-running slide show of your hi-res pictures. The program uses page flipping so the viewer never sees the pictures load.

```
10 TEXT:HOME
20 L1=8192:L2=16384:LOC=L1
30 READ F$
40 IF F$="DONE" THEN GOTO 500
50 PRINT CHR$(4);"BLOAD";F$;",A";LOC
60 C=C+1
70 IF C=1 THEN POKE -16297,0:POKE -16302,0:POKE -16304,0
80 IF LOC=8192 THEN POKE -16300,0:LOC=L2:GOTO 100
90 POKE -16299,0:LOC=L1
100 GOTO 30
500 END
1000 DATA mypic1.pic
1010 DATA mypic2.pic
1900 DATA DONE
```

Put your picture names in the data statements beginning at line 1000. Don't type "mypic1.pic";do type the name of your files exactly as they appear in the catalog. Don't change line 1900; the last name in the DATA list must be DONE.

The key to this program is in the use of a variable to represent where the picture is to be loaded. The variable LOC contains either 8192 (page 1) or 16384 (page 2). The IF statement in line 80 is responsible for flipflopping the value of LOC. Line 50 loads a picture to the value stored in LOC. When the program is RUN, LOC is set to 8192 and the first picture is loaded to page 1. Line 60 contains a counter so that the necessary screen switches will be thrown to display hi-res graphics the first time the program is run. Line 80 does the POKE to turn on page 1, changes LOC to 16384 and jumps to line 100. Line 100 jumps back to the READ statement to find out which picture to load. LOC is now 16384, so the picture is loaded to page 2 but we're still looking at page 1. Now that LOC is 16384, the IF in line 80 is false, so BASIC falls through and executes line 90. The POKE in line 90 turns on page 2 and changes LOC back to 8192. And so on .... until BASIC READs an END (DONE) from the DATA statements.

What's a wipe?

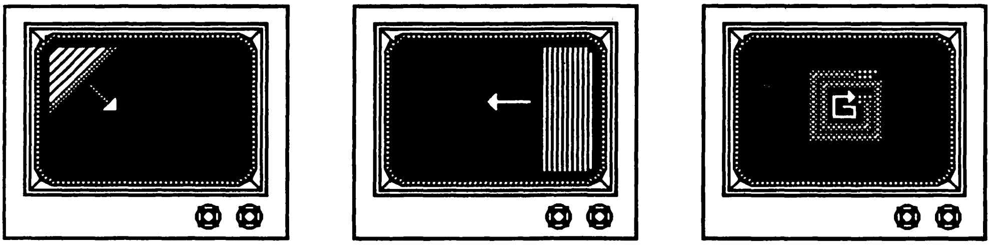
A wipe means to literally wipe off one picture and replace it with another. It's a term derived from video technology, and refers to a transition between pictures.

When you BLOAD a graphic into the Apple computer, the image comes onto the screen, relatively slowly, in sections. (See Why do pictures load like venetian blinds? p. 164) Generally speaking, this is a rather unattractive way to present your graphics to a viewer.

Using machine language routines, programmers have developed a number of alternate wipes. These are usually effected by loading the image, in the standard way, onto hi-res page 2 as the viewer is looking at hi-res page 1. The image is then moved (wiped) onto hi-res page 1. Some wipes have the image scroll into view from the top,bottom, left, right, or corner of the screen, some have the image radiate out from the center of the screen, and some have the image appear in pieces, like a mosaic. The speed of these wipes can usually be controlled.

While very effective wipes have been created, one—the dissolve—still eludes us. A dissolve refers to the technique of fading out or fading in an image by having the image decrease or increase in intensity. The nature of Apple computer graphics does not allow for this.

Slideshow programs usually provide a variety of optional transitions or wipes.



Can I trace or transfer a drawing onto the screen?

There are a few ways that you can trace or transfer an image from paper or fabric onto the computer graphics screen.

One way would be to photostat the original onto a sheet of clear acetate and tape it to the CRT screen. You could then use any of the graphics input devices to trace this image. The light pen would probably be the most accurate tool for this, as you can place it directly on the drawing as you work. You could do the same thing using a graphics tablet or touch tablet—you'd be watching the screen as you directed your hand on the stylus.

Another way to trace an image is to tape the picture to the graphics tablet and trace it with the stylus, or tape it to the table and trace it with another device. You can do this with the touch tablet if you're working with a picture that's small enough to tape onto the touch tablet's surface. A product called Mouse-Around is one of a few devices that adapts a mouse for tracing. The mouse fits into a horseshoe-shaped piece of plastic. At the edges of the plastic are etched crosshairs. As you move the mouse, you look through the plastic, guiding the crosshairs over the drawing.

There's a peripheral called the Digital Paintbrush system that is similar to but a bit more sophisticated than the Koala Pad. It has a ballpoint pen attached to it and traces well. The pen point can be replaced with a plastic stylus for tracing over fabric or pictures that you don't want to mark up.

Another hardware device is the Versawriter. Instead of a stylus it has a plastic extendable arm attached to it with a small see-through lens at the tip. It works very much like a pantograph. It's advertised as a freehand drawing tool, but it functions best when used for tracing.

The most sophisticated and the most expensive tool for transferring images to the screen is a digitizer. With a digitizer you can transfer drawings, photographs, and three-dimensional objects. (See What's a digitizer? p. 246)

How do I create a title screen?

A computer graphics artist is frequently commissioned to create title screens. Whether the screen is for an adventure game, an educational program, a business presentation, or a resumé disk, the goal is the same—create an attractive screen, combining text and graphics, that relates to a subject or theme.

Designing a title screen involves the same concepts as designing posters, book jackets, and so on. Keep in mind all the principles of good design that you have learned and applied to more traditional graphics media.

When you design screens you're working within a tight area. Keeping this in mind, select a font that's small enough to use the space well. Let the title breathe—provide adequate margins.

The huge selection of fonts available for the Apple computer may tempt you to use overly ornate hard-to-read styles of lettering. Don't. Flourished fonts and gimmicky fonts have their uses, but generally a title screen is not one of them.

Consistent spacing between letters and words is important. Use proportional fonts. The traditionally accepted spacing between words is the width of the letter c or o. Letters should be spaced so that, ideally, two straight letters such as ML are farthest apart, two curved letters such as OC are closest, and straight letters next to curved letters such as NC fall in-between. To the eye, open spaces between letters should look the same.

A key word in the title may be emphasized by having it differ from the rest of the words in size, weight, style, or color. This is done easily with the computer.

The illustration must enhance the title, not compete or detract from it. Don't try out every computer graphic special effect on one screen. Keep your graphics clean. Eliminate errant pixels and avoid color combinations that may cause problems.

You'll need a text-generating program that lets you load in your graphic and type directly on it. Choose a program that gives you well-designed proportional and nonproportional fonts as well as control of spacing between characters and lines. Remember, if your letters are to be displayed in color, be sure to select a font with vertical lines at least two pixels wide (for double-hi-res, four pixels wide.)

How do I print graphics?

First, the printer must have graphics capabilities. Many older printers and most letter-quality printers can't print graphics. Given a printer that can, there are two basic ways to print or dump the graphics screen. Both methods require a program to translate the Apple's graphic screens into printer commands—the difference is in where this program is.

Many printer interface cards can print graphics using keyboard or program commands. The graphics dump program is stored in ROM on the printer interface card. There's little standardization among interface card manufacturers—each has unique command characters. Assuming your printer is in slot 1, here are the key sequences used by two popular cards. (You can use the command PRINT CHR$(9) instead of CONTROL I.)

Orange Micro Grappler interface:

PR#1 RETURN	Activate printer in slot 1
CONTROL I G RETURN	Dump screen hi-res 1
PR#0 RETURN	Return output to screen

Interactive Structures Pkaso interface:

PR#1 RETURN	Activate printer in slot 1
CONTROL I H RETURN	Dump screen hi-res 1
PR#0 RETURN	Return output to screen

If you're using a different interface card, read the documentation that came with it. Both the Grappler and Pkaso offer options to enhance the "plain vanilla dump" given in the command sequence above. While most interfaces and dump programs print hi-res screens only, some print lo-res and/or double hi-res graphics.

The other way to dump graphics is to use software. There isn't enough room on an interface card ROM to program lots of bells and whistles—not a problem with a dedicated printer dump program, some of which allow you to crop a picture, border it, produce a cameo, add text, and so on.

We use both a graphics interface and dedicated graphics dump software. The direct command method is good for quick hardcopy of graphics for reference purposes; the software gives us high-quality dumps for professional uses.

What are presentation graphics?

Presentation graphics are often referred to as business graphics, but there's a fine distinction between the two. Business graphics refers to charts, graphs and other analytical images that people would create for their own use, while presentation graphics are images designed expressly to convey information to others.

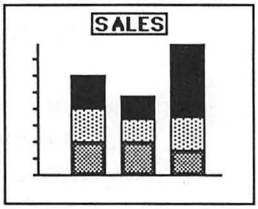
A graphics presentation can be shown to two or three people, to an auditorium full of attendees, or to passing crowds in a department store. The presentation can be self-running or designed for an audience that will interact with the computer via keyboard, light pen, or touch screen.

The finished graphics can be seen on the computer, they can be converted to 35 mm slides or video, or they can be printed out and presented in paper form.

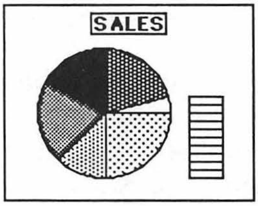
Presentation graphics created with computers are in great demand. These graphics are used by speakers, trainers, teachers, anyone who needs an effective way to present information such as trends, sales, profits, and so on. Computer publications are filled with advertisements for new software and hardware that offer more and more elaborate options for the creation of bar charts, pie charts, animations, and other graphics suitable for presentations.

The sales pitch for these products emphasizes a do-it-yourself approach to the executive, but few executives have the time or the inclination to create graphics. Besides, no matter how easily the software can be used, the artist's touch will always make the difference between high-quality and passable graphics.

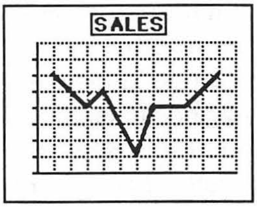
Presentation graphics



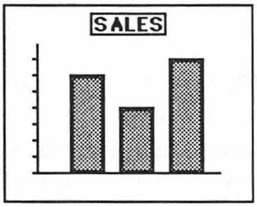
Stacked Bar Chart



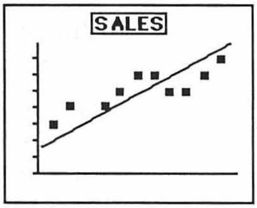
Pie Chart



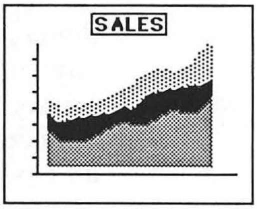
Line Chart



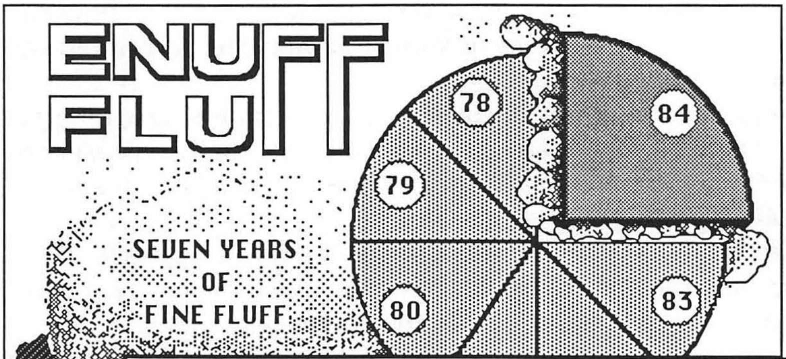
Bar Chart

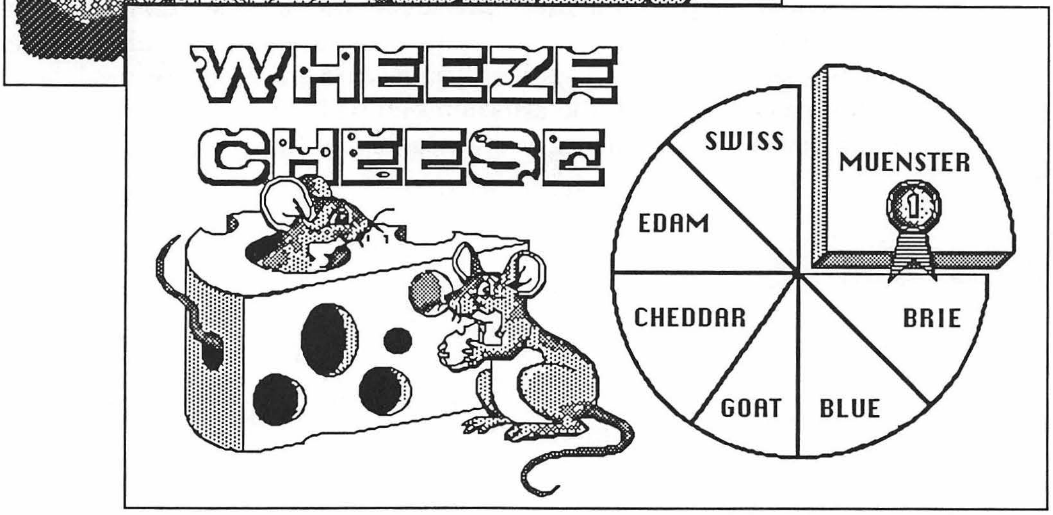


Scatter Chart



Area Chart





Pie charts with a humorous approach

How do I create presentation graphics?

The guidelines for effective computer presentation graphics are the same as for any commercial graphics.

Before beginning the images, plan the entire presentation. Ask yourself: What's my objective? What do I want to present? Analyze the intended audience. Make an outline of the material to be covered and decide whether a serious or humorous approach would be more effective. Create a storyboard.

The storyboard, a sequence of sketches of each visual in the presentation, will serve as your guide until the project is completed. It's better to have many simple screens than a few complex ones. Don't distract the viewer with a visual that's cluttered with symbols and/or words; limit each image to one main idea. If you include screens of text limit the number of words on each screen to a maximum of fifteen, and don't get carried away with decorative fonts.

While any good paint program can be used for creating your images, you might find that you're dealing with lots of numerical data. If this is the case, you want a good business-graphics package which specializes in the creation of labeled charts and graphs from raw data. The program should give you a choice of graph type: bar, pie, or scatter/line graph. If your presentation involves scientific data you may need a program that specializes in scientific plotting. If you need to create charts such as flowcharts, floor plans, or circuit diagrams, a charting package may fit the bill—all are availiable for the Apple II family of computers.

When you're called upon to design and create a presentation, consider a range of approaches. Simple animations can spice up an otherwise dull presentation. Sometimes a bar chart is more effective if, instead of traditional bars, the product is illustrated. Car sales, for example, can be shown with a chart made up of cars. Shape tables are excellent for this application.

When designing presentations for a client, be prepared to suggest alternative ways of presenting the visuals. Become familiar with slide show utilities that present computer graphics with various transitions between images. Know the pros and cons of converting images to slides. Learn about different printer and plotter utilities for printed output. Other options include overhead transparencies, large screen projection systems, video tape, and video disk. Your client will appreciate your knowledge and value your suggestions.

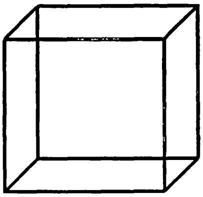
What are three-dimensional graphics?

A three-dimensional object is similar in concept to an Apple shape. The object is composed of vectors—lines that connect points. Unlike Apple shapes, however, the 3-D object has depth— a Z axis. A two-dimensional object, like an Apple shape, can be thought of as a three-dimensional object without depth.

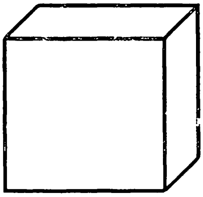
While mathematics has imposed strong conventions on the X and Y axes, the Z axis is less often used. 3-D computer graphics may be accomplished on a left- or right-handed coordinate system. These terms refer to the orientation of the Z axis. If this page were the X and Y surface, the Z axis of a right-handed system would be positive as it extended toward you and negative going away from the other side of this page. Left-handed systems are just the opposite; the Z axis would be negative toward you and positive extending away from the back of this page. Both types of systems are available for the Apple.

3-D objects, like 2-D shapes, can be rotated, translated, and transformed (scaled). Unlike Apple shapes, 3-D objects are not supported by Applesoft. To work with them, you'll need a 3-D graphics package.

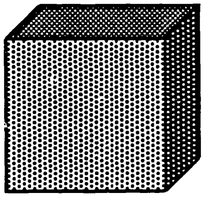
No Apple 3-D system that we know of performs one of the most important three-dimensional manipulations: hidden line removal. In hidden line removal, those parts of a 3-D object that can't be seen from the viewer's point of view aren't drawn. Similarly, no Apple system can perform shading of the 3-D object—objects are seen only as "wire frames." Hidden line removal and shading require extensive mathematical manipulation, which is impossible for a microcomputer to do in real-time. You may, however, save a screen image of a 3-D object in various stages of rotation or transformation and create a solid object by coloring it in a paint program.



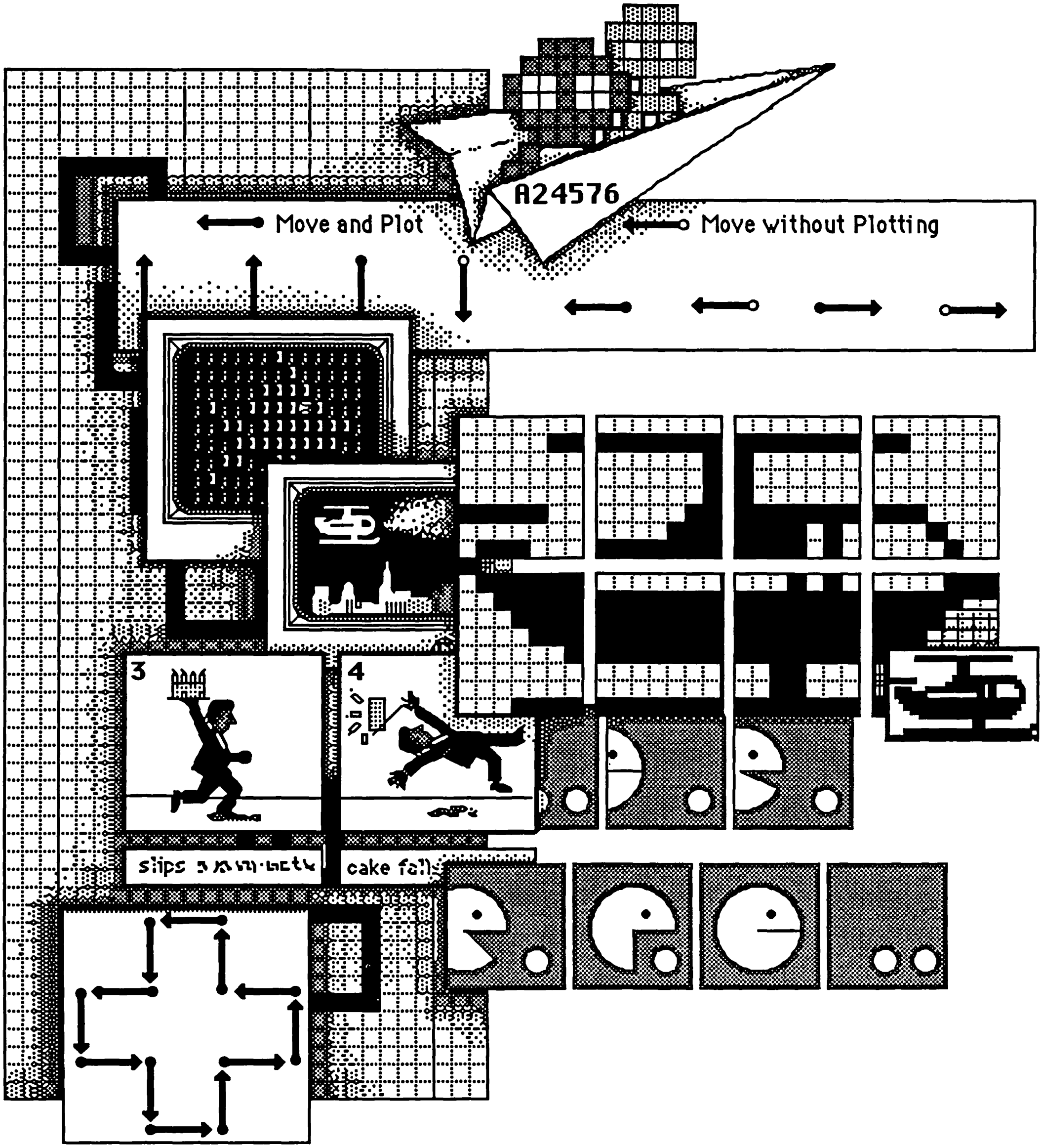
wire frame



hidden line removal



hidden line removal and shading



About animation

About animation

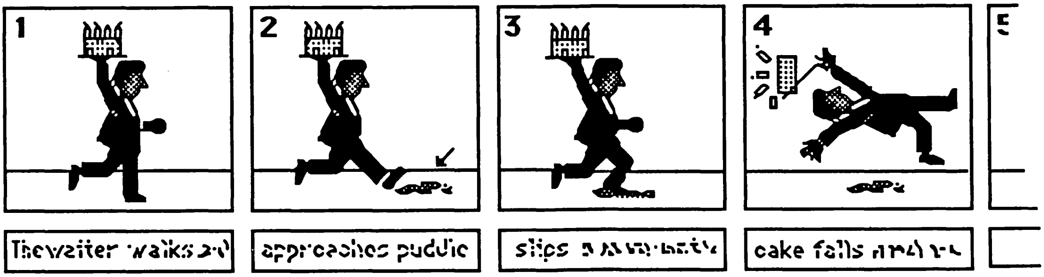
What's animation?

Animation, for our purposes, can best be described as movement. It involves the simulated movement of drawings and shapes. Simulated is the key word here, for in actuality there is no movement.

Animation is an illusion of movement that can be created because of the phenomenon of "persistence of vision" in which the eye retains an image for a split second after that image has been removed from sight. In an animated film, an image drawn frame by frame, its position altered slightly in each frame, appears to be moving when the frames are sequentially projected at the appropriate speed.

When we think of animation, most of us think of Walt Disney, who, perhaps more than any other individual, established animation as an art. Disney animation is cel animation—the images are drawn onto a series of transparent sheets made of cellulose acetate, called cels. Using cels instead of paper, the artist doesn't have to draw in the background and foreground every time there's a change in position. Where only a small part of the image needs to move between frames, the new action is drawn onto a separate cel.

The perfectly registered, hand drawn and painted cels are placed on top of each other on an animation stand and photographed, frame by frame. The final film consists of thousands upon thousands of frames. Projected at the correct speed, with the aid of the human eye, the images appear to be moving—animation.



Can the Apple be used for animation?

Definitely. But, because of the Apple's limited memory and CPU speed, it's impractical to attempt full-screen animation other than page flipping, which restricts you to two frames.

Most Apple animation uses shapes or blocks of data moving over a stationary background. These represent the two fundamental types of computer animation: vector and block or raster animation. They're differentiated by the display technology used.

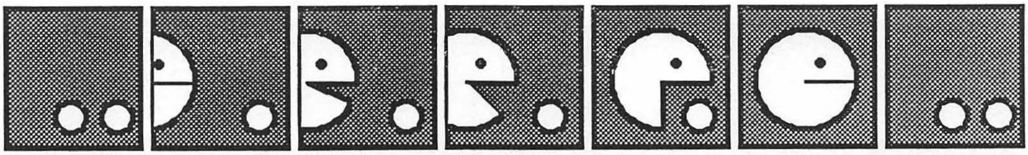
A vector is a line; it has direction and length. Vector graphic systems draw lines. In their displays, the electron gun at the rear of the video tube traces only the graphic—it never traces the blank areas of the screen. The most common vector graphic display is an oscilloscope; the arcade versions of Asteroids and Battle Zone have vector graphic displays.

Rather than a vector display, the Apple has a raster display: a grid of pixels created by an electron gun scanning horizontally across the screen—tracing the entire screen—turning on the pixels for the graphic.

Apple shapes, however, are vector objects. When a shape moves, the vectors are drawn and redrawn. Shapes are often used for animation because they're supported by Applesoft BASIC. Most games, however, use block shapes for animation. Block shapes consist of pixels arranged in a rectangle. When a block moves, the entire block is drawn and redrawn.

The simplest Apple computer animation consists of an object moving across a background. Complex animations feature movement within the object: eyes blinking, legs moving, propellers rotating, and so on. Whether simple or complex, all animation uses the same basic animation cycle:

1. Draw
2. Erase (using background color or design)
3. Change coordinates
4. Goto 1



How do I do computer animation?

All the basic principles of animation apply to computer animation. There are excellent programs available for the Apple computer that provide you with the utilities you need, but remember that an important part of your work is the planning.

Once you've chosen the theme of your animation, decide on the number of "characters" or shapes you'll need. Draw the key shapes on grid paper. Write a short script and then storyboard the animation. Strip everything down to its simple components.

Split actions into sections—animators refer to this as key positons. Once these positions have been decided on, the intermediate actions are drawn in. Subtle changes in the positon of objects result in smooth actions. The more you change the object's position between screens (or frames if you're creating cel-like animation), the more jerky the action will appear.

Each portion of action, from one key position to the next, is known as a phase. When the phase from one key position to the next repeats itself, it's called a cycle. For example, the feet of a person walking will return to the same position at regular intervals. Look for duplicated cycles in the action. If you're simulating realistic activities, find out how long the action you're depicting takes—time yourself as you imitate the movement.

Avoid complicated shapes and movements. The resolution of the graphics screen and the idiosyncrasies of Apple's color necessitate that you keep your images simple.

Experiment with these techniques:

Use distortion to create an illusion of movement. To get the effect of speed, elongate the image away from the direction of movement. Show loss of speed by compressing an image; for example, draw a bus curved into half its normal size with the back wheels meeting the front wheels.

Transform one image into another. Turn a building into a person. The effect is produced by creating a series of shapes, gradually changing the shape with each drawing.

Have a drawing draw itself. Draw part of an image, save, add to the image, save, and so on until the image is complete. Then, run all the screens in a slide show. Have the drawing undraw itself by reversing the order of the screens. (This effect is created easily with the Picture Painter utility in the Graphics Magician.)

Can the text page be used for animation?

Yes, you can use BASIC commands to animate all the text page characters—letters, numbers, and symbols.

VTAB and HTAB are the keys to text-page animation. These commands are used to position the cursor to a specified PRINT position. Here's a program to illustrate simple text animation:

```
10 HOME
20 M$="Graphics by NAME"
30 H=20-LEN(M$)/2
40 FOR I = 2 TO 20
50 VTAB I: HTAB H: PRINT M$
60 VTAB I-1: HTAB H: PRINT SPC(LEN(M$))
70 NEXT
```

Put your name in line 20. Line 50 prints the message, line 60 erases it, and the change in VTAB is taken care of by the FOR/NEXT loop in line 40. That's it, print-erase-print, the animation cycle.

To make this program work with a patterned background, make the following additions and changes:

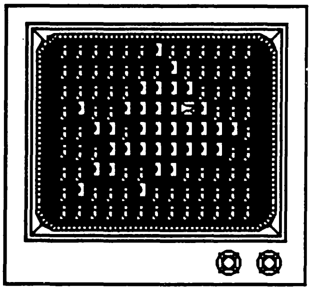
```
15 FOR I = 1 TO 24: FOR J= 1 TO 40-(I=24)
16 PRINT ".";: NEXTJ: NEXT I: POKE 2039,174
25 FOR I = 1 TO LEN(M$): B$=B$+".": NEXT
60 VTAB I-1: HTAB H: PRINT B$
```

Lines 15 and 16 fill the screen with periods. A little trick in line 15—"40-(I=24)"—assures that only 39 periods will be printed on line 24. The POKE in line 16 puts a period in the lower right-hand corner of the screen. (See How do I position text? p. 113). Line 25 creates a string of periods as large as the length (LEN) of M$ and assigns it to B$. Line 60 now prints this string of periods rather than a blank. The field of periods is preserved. (For more sophisticated text animation see Graphic Bonus: Text Animate, p. 181.)

Graphic Bonus: Text Animate

Use SHIFT M to type "]" on the Apple II and IIplus.

```
 5 SW=40: REM Screen width: 40 or 80
10 A$(1)=";;;;;];;;;;"
20 A$(2)=";;;;;;];;;;"
30 A$(3)=";;;;]]]];;;"
40 A$(4)="];;]]]]6];;"
50 A$(5)=";]];]]]]]]]"
60 A$(6)=";;]]]]]]]];"
70 A$(7)=";]];;]];;;;"
80 A$(8)=";];;;];;;;;"'
90 A$(9)=";;;;;;;;;;;"
100 FOR I=1 TO 24
110 FOR J=1 TO SW-(I=24)
120 PRINT ";";: NEXT J: NEXTI
130 POKE 2039,187
140 H=1: V=10
150 FOR I=1 TO 9: HTAB H: VTAB V+I
160 PRINT A$(I): NEXT I
170 FOR I=1 TO 9: HTAB H: VTAB V+I
180 PRINT A$(9): NEXT I
190 H=H+1: IF H>SW THEN H=1
200 GOTO 150
```



You'll have to press CONTROL C to stop this program. There are two loops (lines 120-130 and 140-150); the first draws and the second erases. There's considerable screen flicker because you see the erase cycle—there are moments when the screen is all background. To eliminate the flicker we can make the draw cycle do double duty (erase as it draws) by adding an extra semicolon on the left side of each string. Insert this line:

```
95 FOR I=1 TO 9: A$(I)=";"+A$(I): NEXT I
```

We no longer need the erase cycle, so delete lines 170-180. With this extra column of semicolons, the animation now erases itself as it draws. Since you never see a totally blank screen, the animation is flicker-free.

How do I animate on the lo-res screen?

Because the lo-res screen is in the same area of memory as the text screen, you can use BASIC text commands to animate on the lo-res page.

Start with the program from Graphic Bonus:Text Animate, p. 181, and add the following line:

```
2 POKE-16298,0: POKE-16304,0: POKE-16302,0
```

When you RUN it now, you'll see the same fish you saw on the text screen but in lo-res graphics. We chose the bracket and semicolon for this animation because they translate into solid color lo-res blocks. Only 16 characters in the Apple character set display this way. The others, like the 6 that's used for the fish's eye, display as two-color blocks. (See Low-resolution color blocks, p. 131)

Somewhat more sophisticated animation can be achieved using lo-res graphics commands. The key to lo-res animation is in remembering how to erase: plot in the background color. Here's a bouncing ball program:

```
10 GR
20 YI=+1: XI=+1
30 C=INT(RND(1)*16): IF C=0 THEN GOTO 30
40 X=INT(RND(1)*40): Y=INT(RND(1)*40)
50 XS=X: YS=Y
60 COLOR=C: PLOT X,Y
70 COLOR=0: PLOT OX,OY
80 OX=X: OY=Y
90 IF X+X1=39 THEN XI=-1
100 IF X+XI<0 THEN XI=+1
110 IF Y+YI>39 THEN YI=-1
120 IF Y+YI<0 THEN YI=+1
130 X=X+XI: Y=Y+YI
140 IF X=XS AND Y=YS THEN GOTO 30
150 GOTO 60
```

The ball is drawn by line 60 and erased by line 70. The variables X and Y hold the current position of the ball while OX and OY hold the former position of the ball. Try removing line 70; you may like the effect.

How do I animate on the hi-res screen?

Applying the basic animation cycle to the hi-res page is much the same as it is on the text and lo-res pages.

The simplest form of hi-res animation is a page flip. Rapidly switching the display from page 1 to page 2 and back can, with proper screen design, achieve impressive animation effects. (See What's page flipping? p. 184)

With a BASIC program you can plot an image, erase it with the background color, and replot it at a new location. Because there's a significantly greater amount of compution involved in plotting on the hi-res screen than plotting on the lo-res screen, the slowness of hi-res animation from Applesoft may not satisfy you. Simple animations, however, do work. Witness this modification of our lo-res bouncing ball program.

```
 5 SCALE=1: ROT=0
10 FOR I = 0 TO 10:READ T
20 POKE 768+I,T:NEXT
30 POKE 232,0:POKE 233,3
40 HGR:POKE-16302,0
50 IX=2:IY=2
60 HCOLOR=0:DRAW 1 AT OX,OY
70 HCOLOR=3:DRAW 1 AT X,Y
80 OX=X:OY=Y
90 IF X+IX > 279 THEN IX = -2
100 IF X+IX<0 THEN IX=+2
110 IF Y+IY>191 THEN IY=-2
120 IF Y+IY<0 THEN IY=+2
130 X=X+IX:Y=Y+IY
140 GOTO 60
1000 DATA 1,0,4,0
1010 DATA 39,55,55,53,37,37,0
```

This program uses a shape table and Applesoft's DRAW command to create the figure being animated. (See What are Applesoft shapes and shape tables? p. 186) The animation cycle is identical to the cycle in the lo-res program on page 182. If you change the color of the object (in line 70), it no longer looks like a ball. This is due to the way the Apple produces color and is a bane to all Apple animators.

For sophisticated hi-res animation effects, you should use a commercial animation package.

What's page flipping?

Because the Apple has two graphics pages, you can use a page flipping technique to get simple two-frame, whole-page animation.

In page flipping, you use two images that differ slightly from each other. One picture is loaded to page 1 of hi-res graphics and the other picture is loaded to page 2. Then a program cycles through the screen switches, quickly changing the display from page 1 to page 2 and back.

Here's a demonstration program that will show you page flipping. You don't need any hi-res pictures to use this program, as it creates its own.

```
 10 HOME
 20 T=500: REM time delay
 30 HGR: POKE -16302,0
 40 HCOLOR=1: HPLOT 31,30 TO 51,0 TO 51,160
 45 HPLOT TO 31,160 TO 71,160
 50 HCOLOR=2: FOR I = 0 TO 160: HPLOT 140,I TO 200,I: NEXT I
 60 HGR2
 70 HCOLOR=2: HPLOT 140,30 TO 160,0 TO 180,0 TO 200,30
 75 HPLOT TO 200,50 TO 140,130 TO 140,160 TO 200,160
 80 HCOLOR=1: FOR I = 0 TO 160: HPLOT 31,I TO 71,I: NEXT I
100 POKE -16300,0: FOR D = 1 TO T: NEXT D
110 POKE -16299,0: FOR D = 1 TO T: NEXT D
120 COUNT=COUNT+1: IF COUNT > 200 THEN GOTO 200
130 GOTO 100
200 TEXT: HOME: END
```

Lines 40 - 80 create the screens using standard Applesoft commands. The page flip routine is in lines 100 - 130. A counter (appropriately enough called COUNT) is used in line 120 to stop the flip after 200 flips. Experiment with the delay between flips, the value of T, by changing the assignment in line 20 to smaller or larger values. You may also want to change the IF statement in line 120 to show the flip for more or less than the 200 count.

To use the page flipping technique with your own graphics, see Graphic Bonus: Page Flipping, p. 185.

Graphic Bonus: Page Flipping

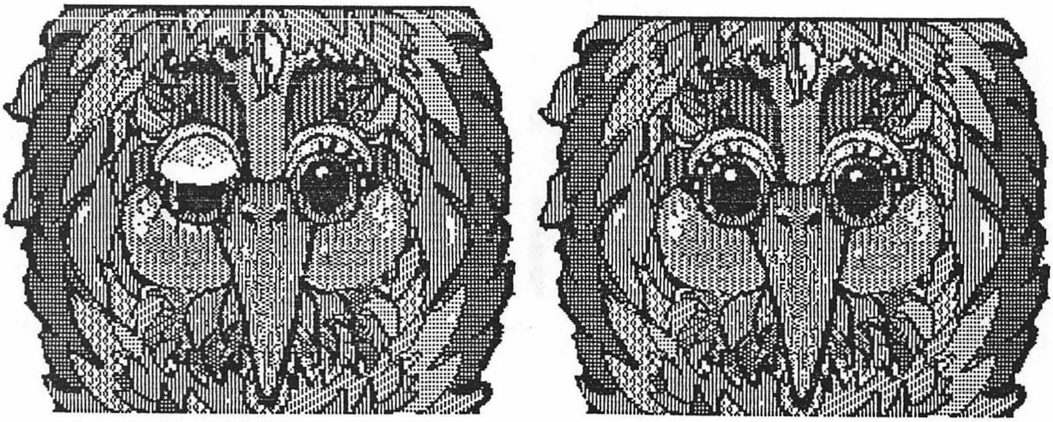
Here's a program you can use with any two high-resolution graphics. Just change lines 30 and 40 so that F1$ and F2$ are assigned your picture file names.

```
10 TEXT: HOME
20 T=500
30 F1$="mypic1": REM put your first picture's name here
40 F2$="mypic2": REM put your second picture's name here
50 PRINT CHR$(4)"BLOAD "; F1$;",A8192"
60 PRINT CHR$(4)"BLOAD "; F2$;",A16384"
70 POKE -16297,0: POKE -16302,0: POKE -16304,0
100 POKE -16300,0: FOR D = 1 TO T: NEXT D
110 POKE -16299,0: FOR D = 1 TO T: NEXT D
120 COUNT=COUNT+1: IF COUNT > 200 THEN GOTO 200
130 GOTO 100
200 TEXT: HOME: END
```

Change the value of T in line 20 to set a delay that's appropriate for your graphics.

Change the IF statement in line 120 to show the flip for more or less than the 200 count.

Finally, you may want this program to lead to another program, or display other pictures, or whatever. You can do this by changing line 200 to accomplish your program goals.



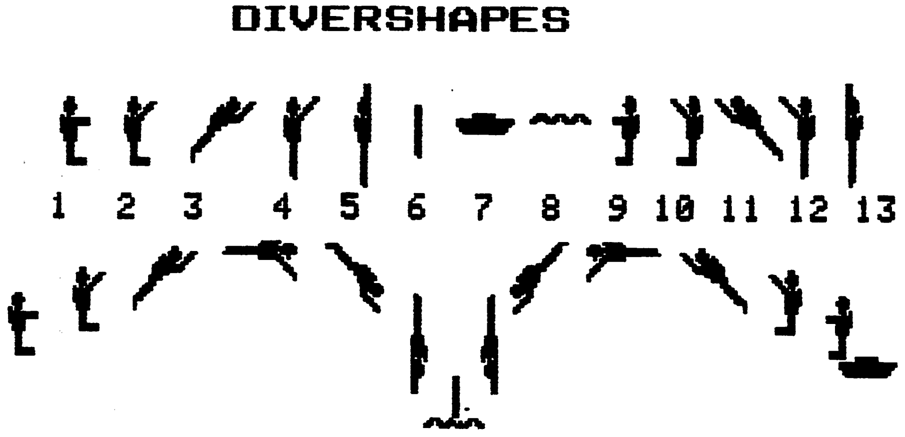
What are Applesoft shapes and shape tables?

Applesoft BASIC allows you to create, draw and manipulate two-dimensional shapes on the high-resolution screen. Shapes can be used for simple animation as well as graphic screens.

Applesoft shapes are vector shapes. A vector shape is made up of lines. To draw a shape, Apple uses a series of lines, or vectors, which tell it whether or not to plot and in which direction to move for the next plot. All the lines are the same length but they may point in any one of four directions—left, right, up, or down. The vectors are placed head to toe until the shape is outlined. A small square, for example, would be defined in this way: plot up 5, plot right 5, plot down 5, plot left 5. A larger square would be: plot up 20, plot right 20, plot down 20, plot left 20. A vector shape looks like an image made by arranging toothpicks.

Once the vectors are defined, they must be "unwrapped," converted to a string of bytes, coded, and stored as part of a shape table. (See How do I make a shape table? p. 193)

A shape table can contain up to 255 shapes, depending upon the sizes of the shapes. Once the shape table is stored in Apple's memory, BASIC statements are used to draw the shapes anywhere, in any of Apples six colors, on either of the two hi-res screens. The shapes can be enlarged (scaled) and rotated.



What can I do with Applesoft shapes?

Applesoft BASIC has four commands which let you manipulate shapes on either of the high-resolution screens.

The DRAW command plots any shape in the shape table, in the last designated color, scale, and rotation, at the X,Y coordinates specified. DRAW 4 AT 20,30 will draw shape 4 in column 20, row 30 of the screen. If DRAW is used without X,Y coordinates, BASIC will draw the shape at the point last plotted by an HPLOT or DRAW statement.

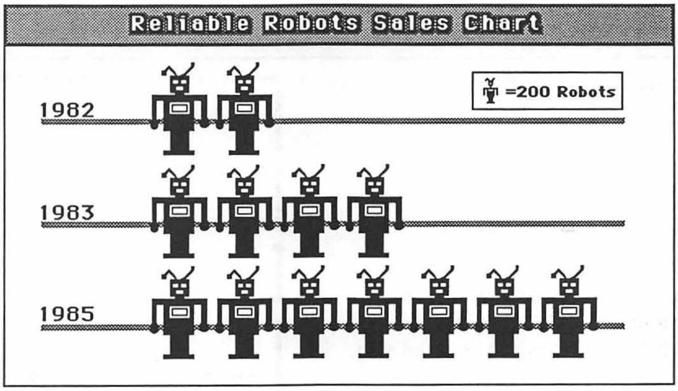
The XDRAW command plots any shape in the table in the last designated scale and rotation, but not in the last designated color. Instead, XDRAW draws the shape in the complement of the color it's plotting over. You can get spectacular effects by XDRAWing shapes in different combinations on the screen. XDRAW can be used to animate shapes. (See How do I animate Applesoft shapes? p. 197)

SCALE changes the size of a shape and ROT rotates the shape.

Applesoft shapes can be used for many applications. When you need a simple image to be used more than once, create a shape. You can, for example, make a "signature" shape and use it to sign your graphics. Some paint programs let you load in your own shape tables to be used from within the program. (The different paintbrushes available in paint programs are shapes.)

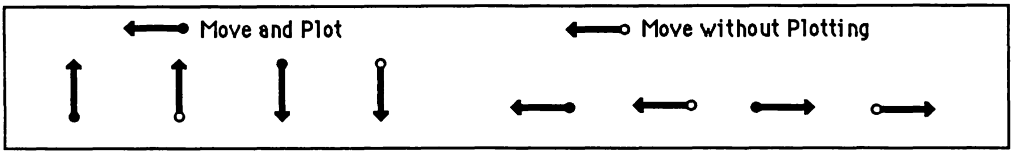
Shapes can be used to create designs and dynamic moving displays as in the Graphic Bonus on p. 198. Use shapes to create unique and effective charts, and to create fonts for lettering on the hi-res screen.

Applesoft BASIC has given you a powerful tool—use it.

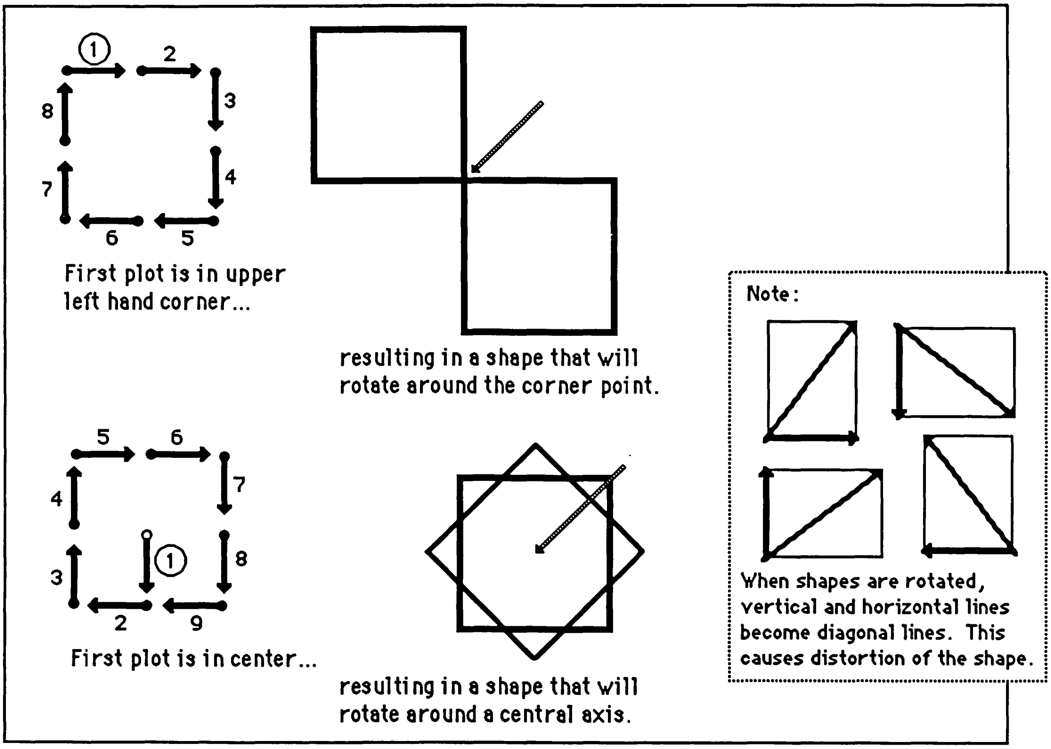


Anatomy of an Applesoft shape

Plotting Vectors



Plotting and Rotation



First plot is in upper left hand corner...

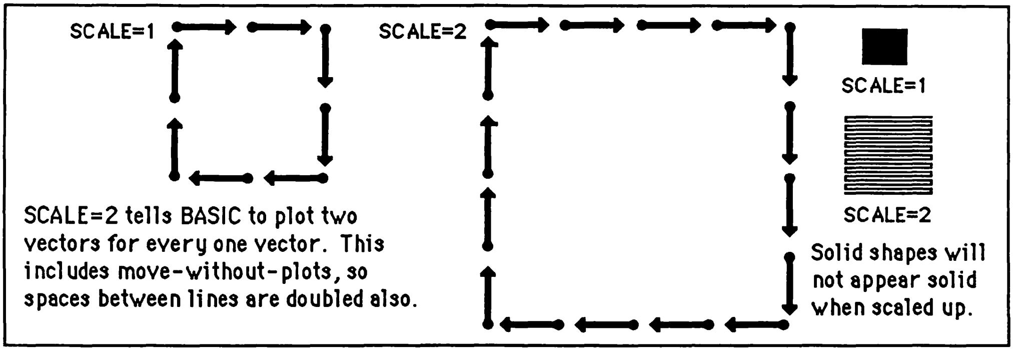
resulting in a shape that will rotate around the corner point.

First plot is in center...

resulting in a shape that will rotate around a central axis.

When shapes are rotated, vertical and horizontal lines become diagonal lines. This causes distortion of the shape.

Plotting and Scale



SCALE=2 tells BASIC to plot two vectors for every one vector. This includes move-without-plots, so spaces between lines are doubled also.

Solid shapes will not appear solid when scaled up.

What's XDRAW?

XDRAW is an Applesoft BASIC command that's used with Applesoft shapes and shape tables. The "X" in XDRAW stands for Exclusive-or (XOR).

Similar to DRAW, the syntax for XDRAW is XDRAW n at X,Y where n is the number of the shape in the shape table. Unlike DRAW, the XDRAW instruction doesn't require a color command—color is irrelevant. When a shape is XORed, the screen pixels are inverted. Where a pixel is on, XOR turns it off; where a pixel is off, XOR turns it on.

The first XDRAW command draws the shape by inverting the unlit pixels on the black screen to white. We see the shape. A second XDRAW command of the same shape, at the same location, inverts the white pixels to black. We no longer see the shape. XDRAW then, is a useful command for animation—draw, erase, draw, erase . . .

Exclusive-or animation has its problems. The inverse of violet is orange or green, and the inverse of green is violet or blue, depending upon whether a whole byte or just one pixel was inverted. You must be careful when you design your shape or the animation will be full of surprises.

If your background is colored, there's no accounting for what will happen when the shapes XDRAW over the different colors. It's best to work on a black background. In fact, if you want a black or white shape on a colored background, you must make the shape the complementary color of the background—do this by plotting pixels in alternate columns. For example. to plot a black shape on a blue background, create a green or orange shape and XDRAW it.

Exclusive-or animation flickers because the erase cycle is being seen. To solve this problem, the animation can use both graphics pages with page flipping. (See What's page flipping? p. 184)

Exclusive-or is not restricted to use with Applesoft shapes, although the XDRAW command is. Many paint programs use XORing with block shapes to achieve special effects. In cut-and-paste moves, for example, the image that's cut or copied is usually XORed as you move it around on the screen. Laying this image over different parts of the graphic creates unanticipated color changes.

How do I SCALE an Applesoft shape?

SCALE transforms or changes the size of a shape. The command SCALE=n, where n can be any number from 1 to 255, specifies the size of a shape when it's plotted with the DRAW or XDRAW statement.

When SCALE is set to 1, the shape is drawn point for point. When SCALE is set to 2, BASIC plots two points for every one plotting vector in the shape, and so on. Note that SCALE=0 is maximum size—BASIC plots 256 points for each plotting vector.

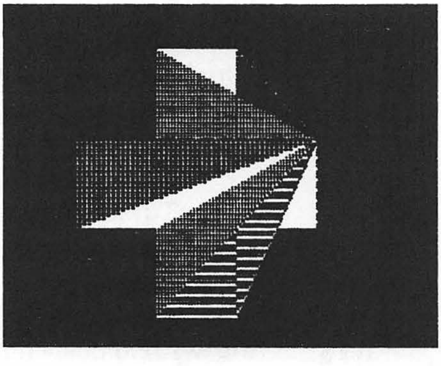
As you scale the shape up, parts of it will eventually go off the edges of the screen, wrap around, and appear on the opposite sides. (See What's wrapping? p. 156) This technique can be used to create interesting effects.

When shapes are scaled, there may be irregularities. If a shape is plotted with multiple lines, for example, the space between the lines is scaled as well as the lines themselves. Therefore, a shape that plots as a solid box when SCALE=1 will become a box made up of lines when SCALE is set to 2 or higher.

Other irregularities occur only if the shape isn't plotted carefully, because every plotting move is magnified when the shape is scaled. For best results see How do I design Applesoft shapes? p. 192.

Try this program:

```
10 HGR: POKE-16302,0
20 HCOLOR=3: SCALE=1: ROT=0
30 FOR I=0 TO 10: READ T
40 POKE 768+I, T: NEXT
50 POKE 232,0: POKE 233,3
60 X=200: Y=75
70 FOR S = 1 TO 50
80 SCALE =S
85 XDRAW 1 AT X,Y
90 XDRAW 1 AT X,Y
100 FOR D=1 TO 200 : NEXT D
110 NEXT S
500 DATA 1,0,4,0: REM HEADER INFO
510 DATA 39,55,55,53,37,37,0
```



Eliminate line 85 and RUN the program again. Eliminate line 100 and RUN the program again.

How do I rotate a shape?

The command ROT=n will rotate any shape in an Applesoft shape table.

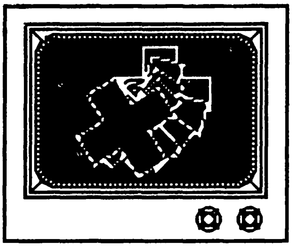
The value of "n" can range from 0 to 255, but in actuality there are only 64 possible rotation settings. When ROT=0 the shape is drawn as it was defined, with no rotation. ROT=16 draws the shape rotated 90 degrees. ROT=32 draws the shape rotated 180 degrees, etc. These rotations repeat themselves starting at ROT = 64.

The number of rotations possible for a shape is dependent upon the scale of the shape. When SCALE is set to 1, only four rotation values are recognized (0,16,32 and 48). When SCALE=2, eight rotations are recognized, and so on. Values that aren't recognized are treated as the next smaller recognized rotation. When SCALE is set to 5 or larger, all 64 rotations are possible.

When a shape is rotated in anything but 90-degree increments, it will be drawn with some distortion. Therefore, if you need a shape displayed at such a rotation, it's best to create a separate shape drawn at that angle.

Try this program:

```
10 HGR: POKE-16302,0
20 HCOLOR=3: SCALE=5: ROT =0
30 FOR I=0 TO 10: READ T
40 POKE 768+I,T: NEXT
50 POKE 232,0: POKE 233,3
55 X=125: Y = 90
60 FOR R = 1 TO 255
70 ROT = R
80 XDRAW 1 AT X,Y
85 FOR D=1 TO 100: NEXT D
90 XDRAW 1 AT X,Y
100 NEXT R
500 DATA 1,0,4,0: REM HEADER INFO
510 DATA 39,55,55,53,37,37,0
```



Eliminate line 85 and RUN the program again.
Eliminate line 90 and RUN the program again.

Next, change the value of SCALE in line 20 to 10, then 12, and so on. RUN the program after each change.

How do I design Applesoft shapes?

Although a shapemaker utility simplifies shape creation, the quality of your shapes depends upon careful planning. Here are some guidelines for "professional" shapes. This list probably won't be helpful to you until you've had some experience working with shapes, so experiment first.

1. Shapes can use up lots of memory, so start by designing your shape on grid paper and plan the direction of the plots. Use as few "plot without moves" as possible to conserve memory. Be aware that BASIC doesn't allow two "move up without plot" directions in a row. Some shape creation utilities compensate for this; some don't.
2. The first vector you define in your shape, your first plot, is the point that BASIC will reference when the X,Y coordinates are specified. If you draw a square shape and you start it at the upper left corner, the command DRAW 1 at 5,10 will plot the shape's upper left corner at X=5, Y=10.
3. The first vector you define in your shape is also the point that BASIC will use as the central axis for rotation. The square shape described above, when rotated, would pivot around that corner point. If you want your shape to rotate like a wheel on an axle, you must start your plotting in the center of the shape.
4. To design a solid shape that will DRAW in color without having to specify HCOLOR in your program, remember how the Apple uses color. If you plot every column your shape will be white, but if you plot every other column of dots your shape will automatically be in color.
5. Since Apple plots colors only in odd or even columns, an HCOLOR other than white may result in a missing vertical line when the shape is drawn. If you intend to plot in both colors of a color set, create your shapes with double vertical lines.
6. When using SCALE and ROT, there may be distortion of the shape. Sometimes it's better to design a separate shape in the desired size or rotation, rather than use the same shape with ROT or SCALE commands.
7. Don't retrace plot lines. If you plot four pixels right and then you want to get to the beginning of the line to plot in another direction, don't replot over the four pixels. The shape may look fine when you DRAW it, but when you XDRAW it, the pixels that were overplotted will create a mess.

How do I make an Applesoft shape table?

For 99% of your shape tables, we recommend that you use a commercial utility to create your shapes. Some programs, such as The Artist, allow you to select (capture) a section of the hi-res screen and automatically create a shape out of it.

Some shapes, especially those which will be scaled up, should be hand-assembled. Such is the case with the shape shown below. At a scale of 1, it looks like a small ball, but at larger scales it looks like a cross. By hand-compiling it, we can assure the integrity of the vectors when scaled.

The steps in making a shape table are:

1. Design shape on grid paper.
2. Decode grid as vectors.
3. Unwrap vectors.
4. Translate each vector into binary equivalent.
5. Combine 2 binary vector descriptions into a byte.
6. Translate bytes into hex then into decimal.
7. Create shape table header.
8. POKE data into memory. (BSAVE if desired.)

Refer to the illustration below. Unwrap the vectors by placing them in a single column. Each vector is described by a three digit binary number as follows:

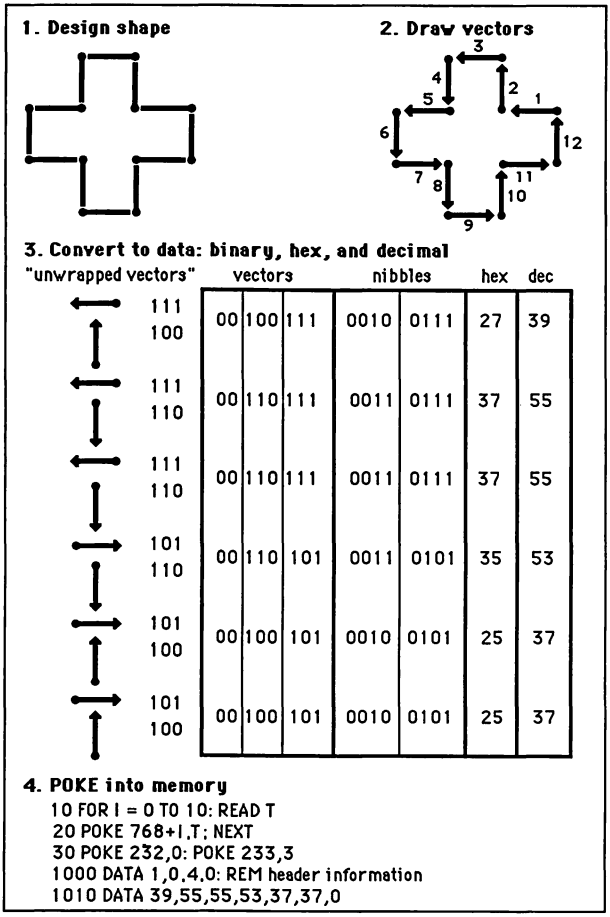
Vector	Binary	Vector	Binary
Move left	011	Draw left	111
Move right	001	Draw right	101
Move up	000[1]	Draw up	100
Move down	010	Draw down	110

Assign the correct binary digit to each vector and combine pairs of vectors together. Pad each pair by putting a 00 in front of the vectors. Now you have a list of bytes, each representing two vectors in your shape. Translate these bytes that define a shape into decimal, (See How do I convert hex into decimal? p. 38), and POKE them into memory.

[1] you cannot have two consecutive move ups.

Before you have a shape table, you must create a shape table header. (See Anatomy of a shape table, p. 195.) The header begins with a byte containing the number of shapes in the table followed by a byte containing a zero. The header continues with a pair of bytes containing offsets—from the start of the table —for each shape definition. For a table containing one shape, the header information is always 1,0,4,0 because the shape definition begins four bytes from the start of the table. For a table containing two shapes (the first of which is 7 bytes long), the header would be 2,0,6,0,13,0. Shape 1 begins right after the six-byte header and the second shape begins after the header and the first shape definition (6 + 7=13).

Creating headers isn't simple for large shape tables. If we need a table with many hand assembled shapes, we create single-shape tables and use a shape table editor (like Pixit) to create the master table.



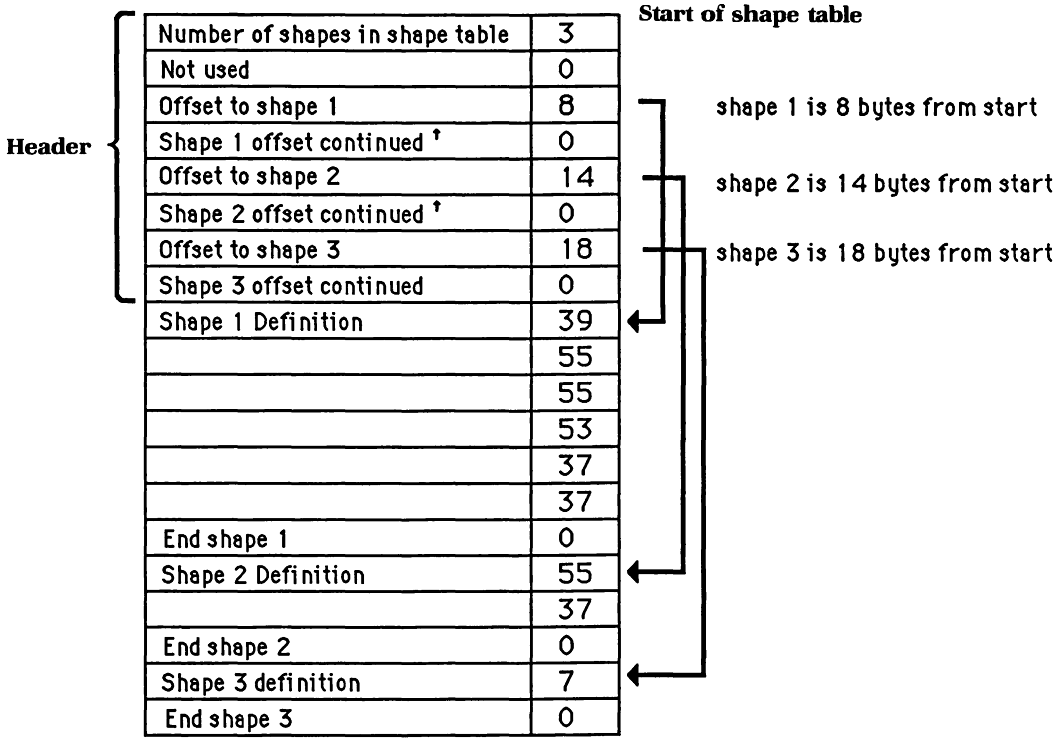
3. Convert to data: binary, hex, and decimal

"unwrapped vectors"	vectors			nibbles		hex	dec
111 100	00	100	111	0010	0111	27	39
111 110	00	110	111	0011	0111	37	55
111 110	00	110	111	0011	0111	37	55
101 110	00	110	101	0011	0101	35	53
101 100	00	100	101	0010	0101	25	37
101 100	00	100	101	0010	0101	25	37

4. POKE into memory

```
10 FOR I = 0 TO 10: READ T
20 POKE 768+I,T: NEXT
30 POKE 232,0: POKE 233,3
1000 DATA 1,0,4,0: REM header information
1010 DATA 39,55,55,53,37,37,0
```

Anatomy of an Applesoft shape table


Start of shape table
Header
shape 1 is 8 bytes from start
shape 2 is 14 bytes from start
shape 3 is 18 bytes from start

Number of shapes in shape table	3
Not used	0
Offset to shape 1	8
Shape 1 offset continued [†]	0
Offset to shape 2	14
Shape 2 offset continued [†]	0
Offset to shape 3	18
Shape 3 offset continued	0
Shape 1 Definition	39
	55
	55
	53
	37
	37
End shape 1	0
Shape 2 Definition	55
	37
End shape 2	0
Shape 3 definition	7
End shape 3	0

[†] offset 2 = number/256
offset 1 = number - (offset2*256)
where number is number of bytes from start

How do I use a shape table?

Shape tables can be used in immediate mode or in deferred mode (from a program). A shape table is a binary file—you must BLOAD it into Apple's memory. For small tables, a limited amount of memory is available beginning at location A768 (A$300). Larger tables can be stored above the second page of hi-res graphics, A24576 (A$6000).

Here's the sequence:

BLOAD the shape table into Apple's memory.

```
BLOAD shape table name, A24576 RETURN
```

Tell Apple where you BLOADed the table by POKEing the address of the table into locations 232 (low part of address) and 233 (high part of the address). Let Apple do the computing:

```
POKE 232, 24576-INT (24576/256) * 256 RETURN
POKE 233, INT (24576/256) RETURN
```

Select a color to plot the shapes in:

```
HCOLOR=3 RETURN
```

The SCALE and ROT commands tell BASIC how to plot the shapes.

```
SCALE=1: ROT=0 RETURN
```

That's it. Now tell BASIC which shape to DRAW and where,in X,Y coordinates,to DRAW it:

```
DRAW 1 AT 20,80 RETURN
```

This program does it all:

```
10 HGR: POKE -16302,0
20 LOC=24576: REM ADDRESS OF SHAPE TABLE
30 PRINT CHR$(4); "BLOAD name of shape table, A"; LOC
40 HCOLOR=3:ROT=0:SCALE=1:REM TRY DIFFERENT VALUES
50 POKE 232,LOC-INT(LOC/256)*256
60 POKE 233,INT (LOC/256)
70 REM ADD YOUR COMMANDS FROM HERE ON
```

How do I animate Applesoft shapes?

To create an animation, we draw the shape, erase it, and then redraw it at a new location. When this is done often enough and quickly enough, the shape appears to be moving.

The XDRAW command draws a shape in the complement of the color already existing at each point plotted. (See What's XDRAW? p. 189) XDRAW, therefore, can be used to erase shapes as quickly as it draws them—animation!

Try this program:

```
10 HGR: POKE-16302,0
20 HCOLOR=3: SCALE=1: ROT=0
30 FOR I=0 TO 10: READ T
40 POKE 768+I,T: NEXT
50 POKE 232,0: POKE 233,3
60 FOR X=10 TO 260
70 XDRAW 1 AT X,100
80 XDRAW 1 AT X,100
90 NEXT X
500 DATA 1,0,4,0: REM HEADER INFO
510 DATA 39,55,55,53,37,37,0
```

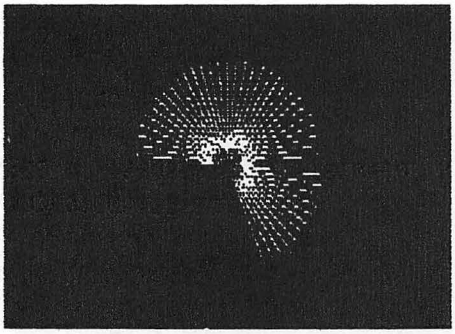
After you've had some experience animating shapes, refer to the shape table animation hints below. Experiment first or these hints won't make much sense to you.

1. When using the draw-and-erase technique, it's best to use XDRAW, not DRAW, even for the initial draw. If you don't do this, the second XDRAW may not erase all of the preliminary draw when rotating and scaling.
2. Simple shapes are drawn and erased quickly, but the drawing of complex shapes may be distracting to watch. In that case, use both graphics pages. While page 1 is being viewed the shape is drawn on page 2 and while page 2 is displayed the shape is drawn on page 1. Using two pages, even for small shapes, eliminates the flickering effect so often associated with Applesoft shape animation.
3. XDRAW animation is easiest and most effectively used on a black background. XDRAWn shapes will appear white or black over color backgrounds if the shape is designed in the complement of that color.

Graphic Bonus: Shape Displays

Simple shapes can generate fascinating displays. One shape that we've used in many programs consists of a single vector. At a scale of 1, it puts only one pixel on the screen, but magnify it and you have magic. Enter the following program:

```
10 FOR I = 0 TO 5:READ T
20 POKE 768+I,T:NEXT I
30 POKE 232,0:POKE 233,3
40 HGR:POKE -16302,0
50 SCALE = 60
60 FOR I = 0 TO 63
70 ROT=I
80 XDRAW 1 AT 140,100
90 XDRAW 1 AT 140,100
100 NEXT I
1000 DATA 1,0,4,0,7,0
```



A student in our class wanted her project to open with a countdown as in the beginning of a film. She drew screens of circles with large numbers inside, and used a program like this to put the rotating line on top of the pictures. In a commercial project we did, a variant of this program provided the rotating line of a simulated radar screen.

Try the program without the second XDRAW by deleting line 90. The colors and patterns generated are a result of the interaction of artifacting and aliasing. Make the shape bigger by changing line 50 to SCALE=255. As the shape wraps, the patterns become quite interesting. Run the program through more than one full rotation by changing line 60 to FOR I = 0 TO 255. Now, the second and fourth rotations erase the shape drawn by the XDRAWs of the first and third rotation.

Combining translation on the X axis is equally impressive. Make the shape small again by restoring line 50 to SCALE=60 and changing line 80 to XDRAW 1 at I,100. A slightly larger scale is also interesting. Change line 50 to SCALE=100. How about restoring the erase by restoring line 90 to XDRAW 1 at I,100? Let's try these variations with a slightly different shape, one with three vectors: move left, plot right, plot right. When drawn at a scale of 1, it produces two pixels. But when it's enlarged . . .

Change line 10 to read FOR I = 0 TO 6:READ T and change line 1000 to read DATA 1,4,0,4,43,5,0 to generate the new shape.

What's a block shape?

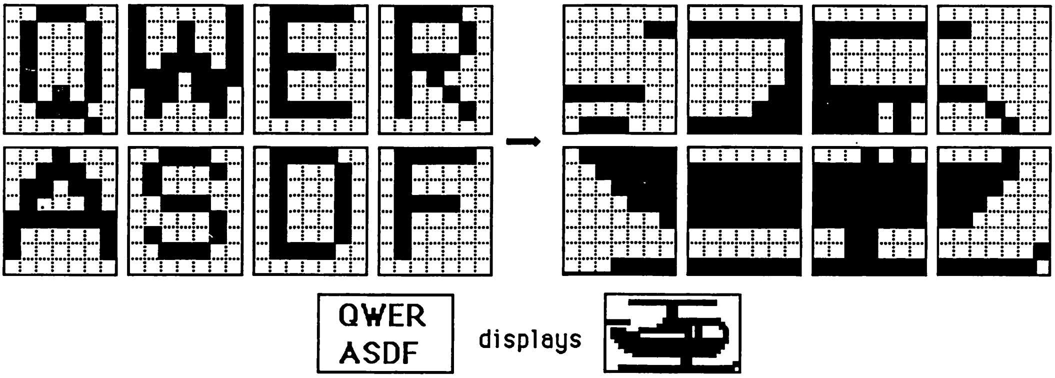
A block shape is a grid of pixels. Applesoft doesn't support block shapes; standard Applesoft shapes are made up of vectors.

The most common way to create a block shape is with a hi-res character generator. (HRCG and Animatrix, on The DOS Tool Kit by Apple Computer Inc. is an excellent utility for this.) Using the character editor that comes with the generator, the artist assigns pieces of a graphic to a character.

The illustration below shows eight block shapes that, when put together, form a helicopter. Each block is made by editing one letter. The eight blocks were assigned to the letters—Q,W,E,R,A,S,D, and F. Each letter, when typed, displays a piece of the helicopter—a program would do the same.

Without the hi-res character generator, typing these letters would produce only the letters, numbers, or symbols that were used to create the shape (in this case—Q,W,E,R,A,S,D and F), but using the character generator, the pieces of the picture would be displayed in place of the letters.

Many graphic programmer's utilities can be used to create block shapes. With these packages, a graphic screen drawn with a paint program can be captured in a series of blocks. Once captured, these blocks are saved to disk for later use.



How are block shapes animated?

One type of block shape animation, character set animation, requires the use of a high-resolution character generator.

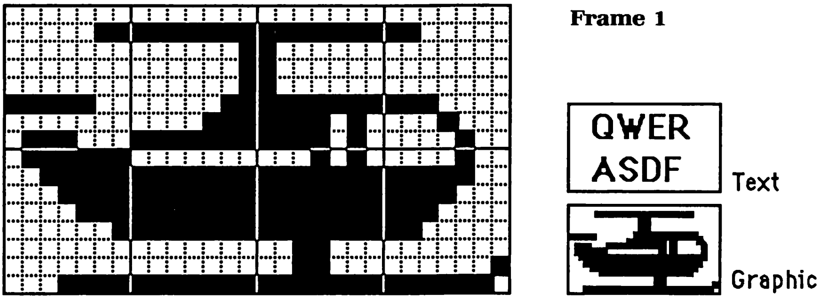
As you assign pieces of a graphic to different characters, you can make subtle changes in each piece. The illustration on the facing page shows four frames of a character set animation. Each frame consists of eight grids of pixels—block shapes. Look at the lower four blocks of each frame and you'll see that there are no changes in the helicopter, so the same letters are used over and over again. The upper portion of the helicopter, however, does change from frame to frame (look at the propeller and tail). Therefore, for each of these block shapes, a different letter is used.

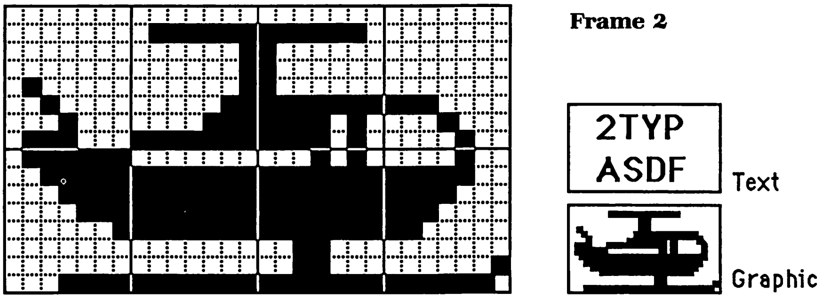
Once the block shapes have been created, an animation program is used to put the pieces on the screen. Drawing and erasing, the program will display each frame in succession. As the frames are cycled through, Frame 1, Frame 2, Frame 3, Frame 4, Frame 1, Frame 2, and so on, the helicopter appears to move. If the program is run without a hi-res character generator, blocks of letters, numbers, or symbols that were used to create the shape will move across the screen, but when run with a hi-res generator, the helicopter will be displayed in place of the letters.

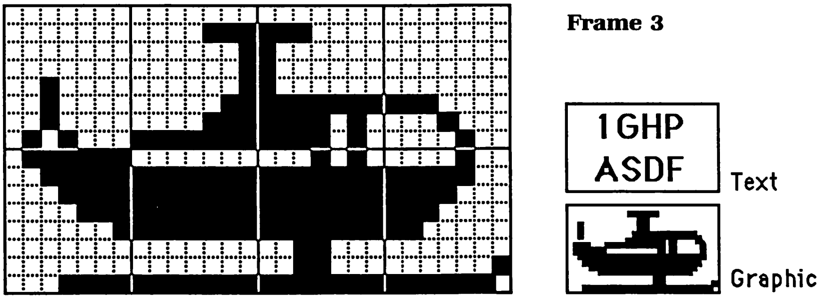
If the helicopter was being animated with HRCG from the DOS Toolkit, the sequence described above would be accomplished by simple print statements.

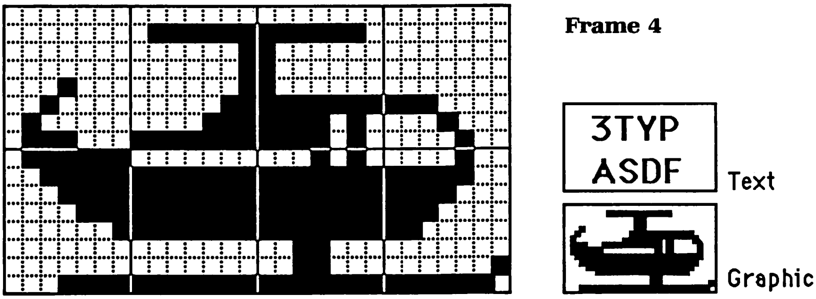
In general, block shapes must be small for flicker-free animation. If the block contains color, be sure the animation program moves the block in steps of two horizontal screen pixels or the colors will alternate between odd and even colors with each move.

Character set animation


Frame 1


Frame 2


Frame 3


Frame 4

Note that the bottom half of the helicopter is the same for all frames, as are other unchanging elements of the graphic.

What's a shifted shape?

Shifted shapes are a special type of block shape. They're essentially premoved animations. For every shape created, seven facsimiles are stored as bit patterns in memory.

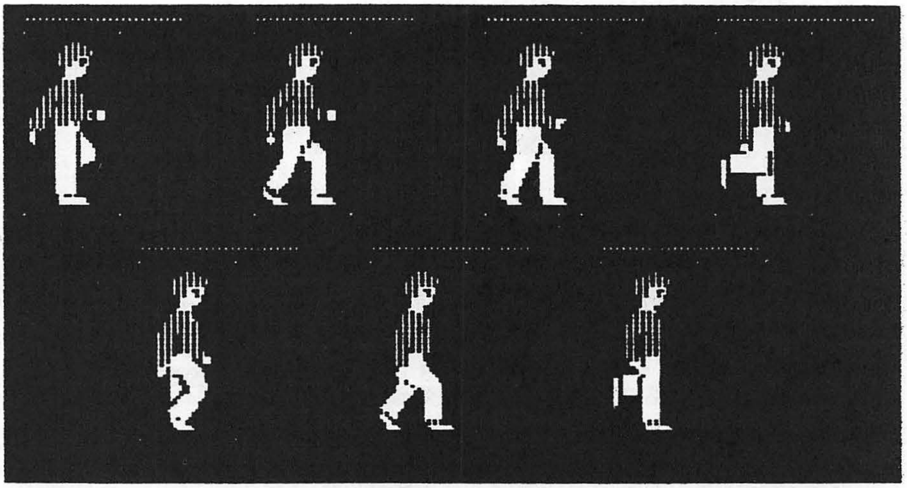
Seven blocks, or frames, are generated for each shifted shape—each frame skewed (shifted) by one pixel from the previous frame. (Because of Apple's odd-even color differences, the frames are usually shifted two pixels to preserve color.)

Each time an animated shape is moved, its new screen position must be computed. The computer does these calculations very fast, but not instantaneously. The calculations are especially slow if the shape isn't being put on one of the 40 byte-boundaries (one every seven pixels) on the high-resolution graphics screens.

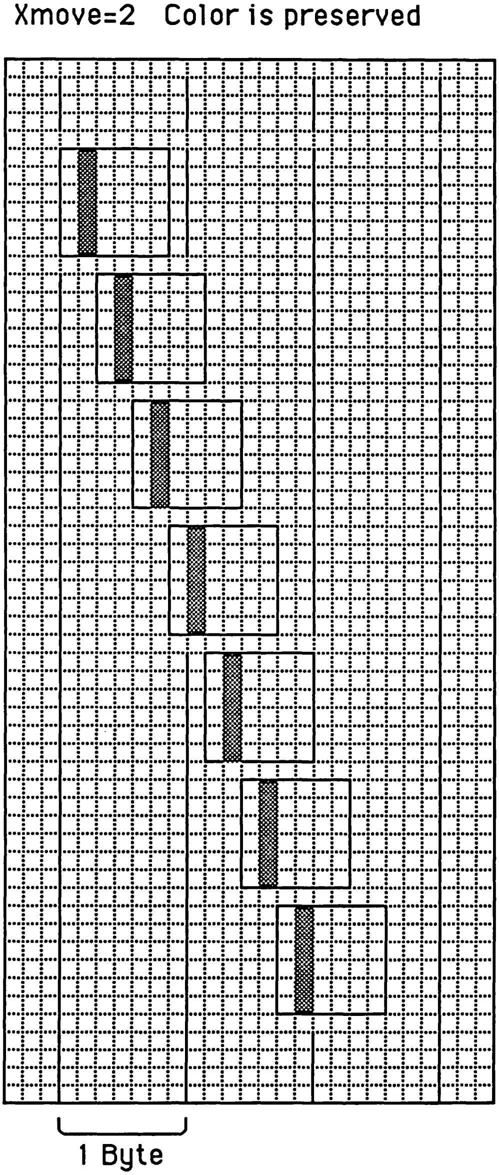
For horizontal movement, new screen coordinates have to be calculated only every seven frames—and then it's always on a byte-boundary. The slowest of the calculations, shifting sideways over two pixels, occur within the shape editor, not within the animation program.

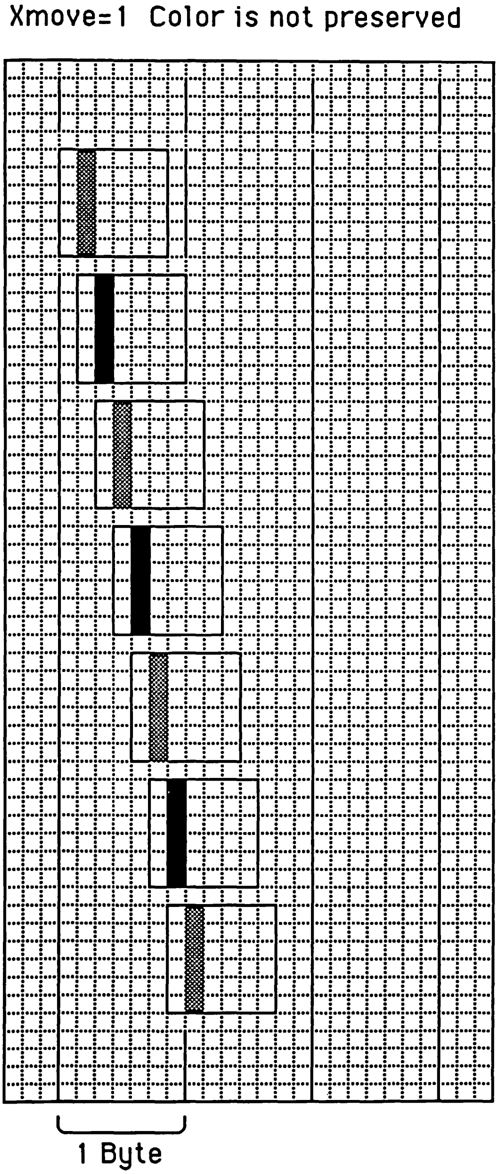
Programs like The Graphics Magician offer routines to use shifted shape animation from BASIC. The animation, once the shapes, paths, and cycles are created with the editors, is accomplished by a simple CALL.

Like standard block shapes, shifted shapes are most effective when they're small. Conservation of background may be a problem and should be considered when designing the animation.



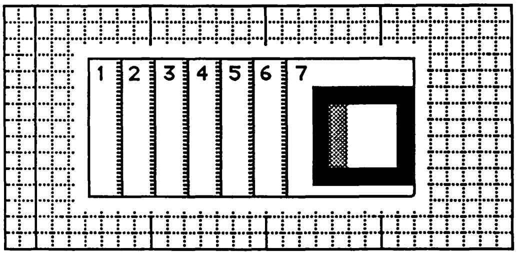
Shifted block shapes



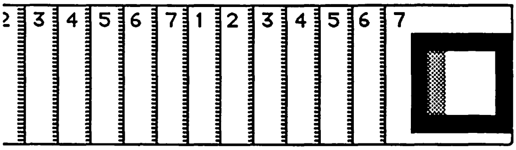


Animating shifted block shapes

Each shift can be thought of as a frame.



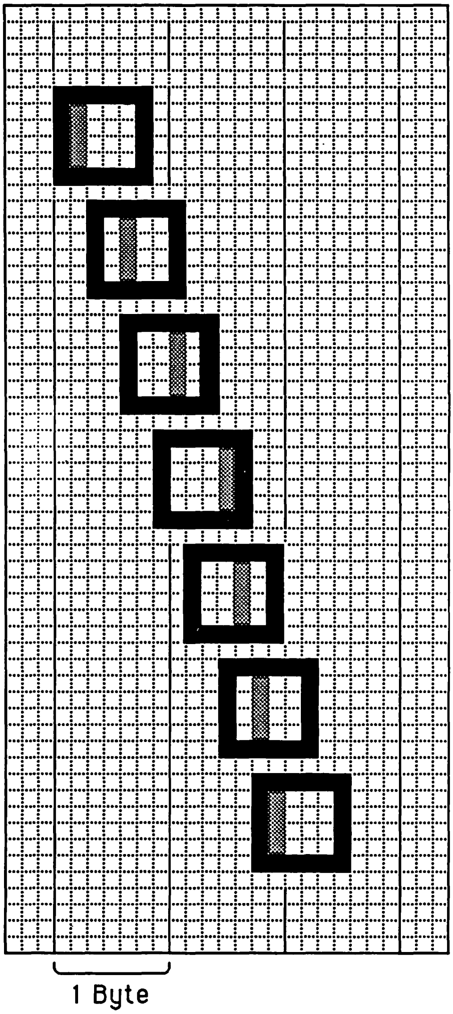
Xmove = 2. Frame 2 erases frame 1, frame 3 erases frame 2, and so on to complete the animated sequence.

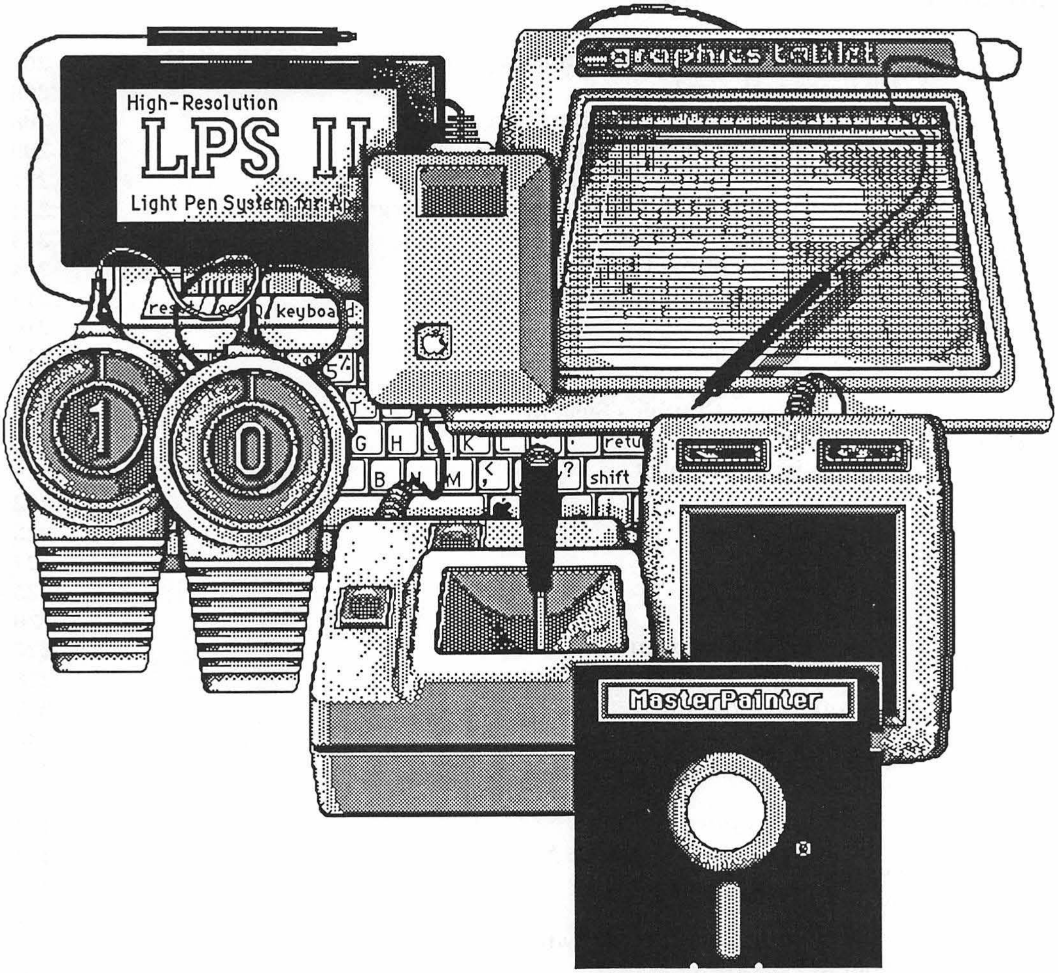


After frame 7, the sequence repeats itself with frame 1 erasing frame 7.

With each shift, the vertical bar moves over one column. The resulting animation would be that of a vertical bar animating internally (moving right and left) within a box.

Xmove = 2 Internal animation





About the artist's tools

About the artist's tools

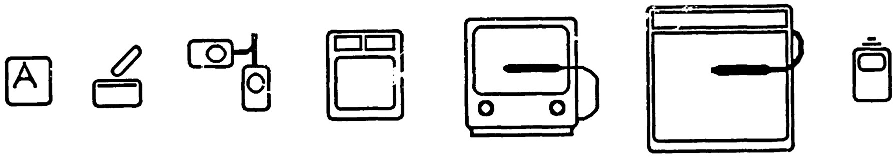
What tools are available to the graphic artist?

Computer graphic tools can be divided into two categories: hardware and software. Hardware encompasses all the graphics input and output devices. (See What do I draw and paint with? p. 139) and software covers the disk-based graphics utilities. (See What kinds of graphics software are available? p. 209)

The range of equipment is staggering—from $50 paddles to multi-thousand-dollar, ultra-hi-res equipment. Add graphics tablets, digitizers, video equipment, printers, monitors, slide-making equipment, and who knows what else?

We can't say how much equipment you'll amass, but based upon our experience, the buying never stops. Just when you think you have everything you need, a new input device or graphics utility comes out. Computers are a relatively new tool, and technology changes so fast that it's difficult, if not impossible, to keep up with the latest advances.

Suffice it to say that just about anything you could want is available for the Apple computer. With it you can create sophisticated images and animations. Although we cover a wide variety of tools in this book, by the time you read it there will be many new products available. The best way to keep up to date is by reading computer publications and attending computer trade shows and computer user group meetings.



Why use software?

You can create graphics on the Apple computer without using commercial software but it'll be difficult and tedious. Applesoft BASIC has graphics commands that let you plot pixels and draw lines in the standard colors, but these commands don't provide many options. To "mix" colors, "paint" with a variety of brushes, or get special graphic effects, you have to be a crackerjack programmer.

Programming is an art in itself. To access many of Apple's graphics capabilities, you must learn machine language. Before you become a computer artist without commercial software, you'll have to dedicate years to becoming a computer programmer.

We believe the programming tool is best left to the professionals. Just as we don't make our own brushes and paper, we don't write our own software. Commercial software provides us with all the tools we need to practice our craft.

How much do I need to spend on software?

Apple computer graphics software varies in price from an average low of $30 to an average high of $80. Most software that costs over $100 includes hardware.

We use a variety of software packages because one program features one option, such as a wide selection of brush shapes and sizes, while another program features something else, such as a better color palette. Then a third program provides a better variety of fonts, and so on.

When possible, try out a program before you buy it. Find out what other computer artists use and why. Assuming that you start with two or three packages, and that a couple of new, enticing programs come out each year, you can expect to spend an average of $200 a year on graphics software.

What kinds of graphics software are available?

The Apple computer has a greater range of graphics software than any other personal computer.

Paint programs

Paint programs give you all the tools you need for the creation of illustrations, title pages, or fine art. These packages let you draw freehand and geometric lines and shapes as well as paint in a variety of brush shapes and sizes. Today, most paint programs offer a varied paint palette and extended editing options. With a good paint program you can magnify or zoom in on an area for pixel-by-pixel touchups; copy, cut, and paste any area of the screen; enlarge and shrink images; exchange or swap colors; and add text to your graphic.

High-resolution character generators

These programs are used for the creation of title screens, charts, and other screens that require varying typefaces or fonts. With most hi-res character generators you can select from a variety of fonts, type anywhere on the screen, change fonts, and vary the font size. A good hi-res text generator offers a variety of proportional and nonproportional fonts, as well as leading and kerning options.

Shape editors and font editors

These utilities are similar to each other, in that both allow you to create specialized files. Shape editors let you create either Applesoft shapes and shape tables or bit-mapped shapes. Font editors allow you to create original fonts that, with the aid of a hi-res character generator, can be used in your graphics.

Animation packages

There are animation packages that help you work with simple Applesoft shapes, or faster, more complex bit shapes. These programs usually provide shape editors where you create the characters that will move, scene editors where you create the backdrops, and "movie" editors where you create the actual animation. Most programs also contain some type of show or projector utility that puts the entire sequence together to present the animation.

Printer dumps

Dedicated printer utilities allow you to get hardcopy printouts of your computer graphics images from a variety of printers—color or black-and-white. Most "dump" programs let you vary the size of your printout, and some have options to border the image, add text to the image, color the image, and so on.

Slideshow software

Slideshows help you make presentations from your computer graphics by loading in a number of different screens and letting you choose to show them automatically (computer-controlled) or manually (the viewer can hit a key or paddle button to move on to the next screen). A good slideshow utility packs the images so that you can get more screens on one disk, and provides a choice of wipes, or transitions, between the images.

Three-dimensional graphics utilities

This software is not as plentiful as other graphics utilities, since it has very specialized applications. Architects, technicians, and designers who need to generate 3-D graphics can find two or three packages on the market.

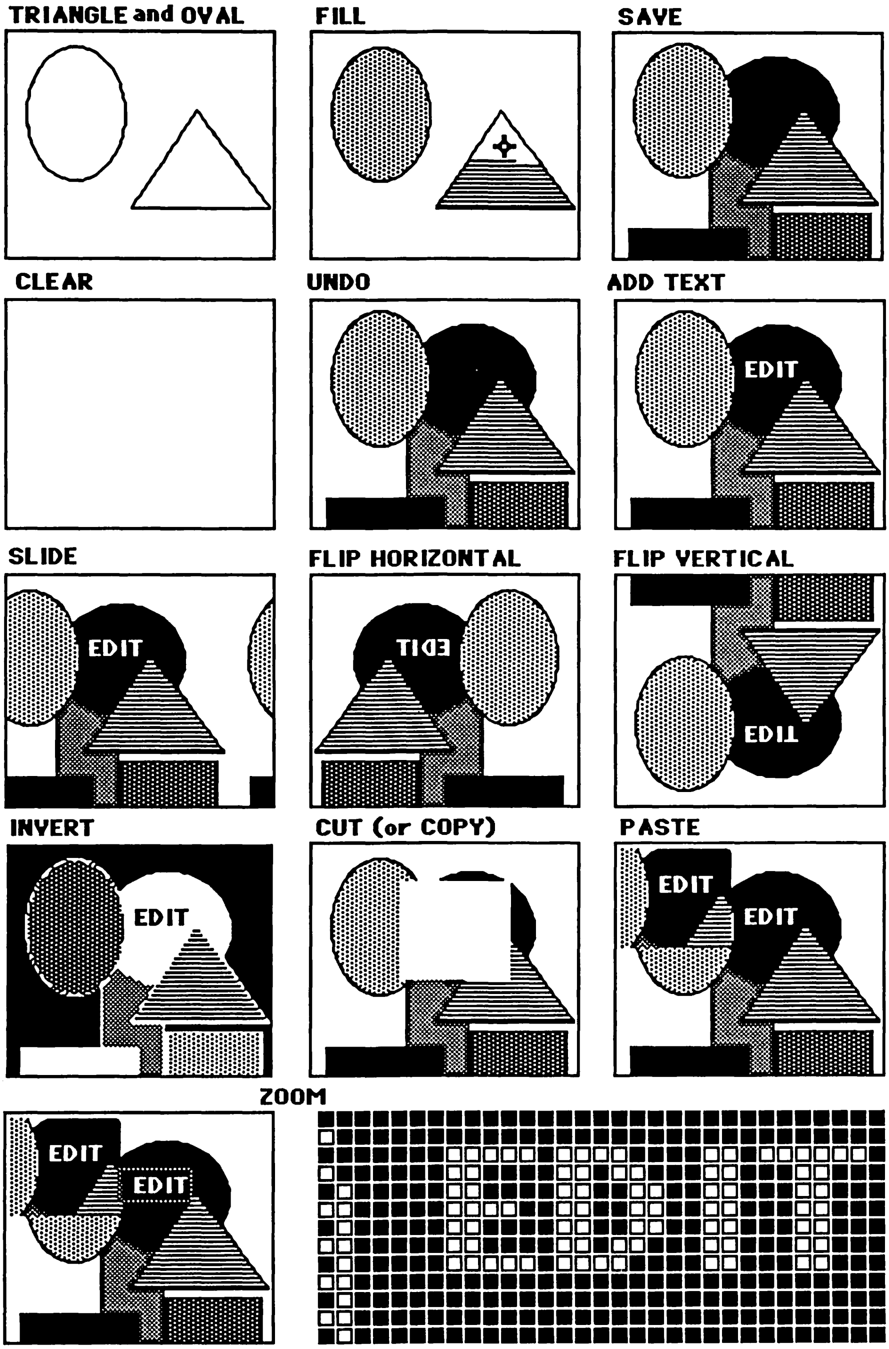
Programmer's utilities

Programmer's utility software provides BASIC and machine language routines that help the programmer get faster, more memory-efficient programs. These utilities will work within BASIC.

Graphics programming languages

Graphics programming languages include Logo, GraForth, and Ceemac. These languages have built-in graphics commands that are not available in BASIC. Logo is known for turtle and sprite graphics, GraForth allows for a number of special graphics options including three-dimensional graphics, and Ceemac is a unique graphics language that uses graphics like musical scores.

Anatomy of a paint program



What makes a graphics software package "good"?

A good graphics utility should meet most, if not all, of the following conditions:

1. The program should do what it says it does. If it advertises a palette of magnificent hues, it should contain more than two versions of Apple's six hi-res colors.
2. The disk shouldn't be copy-protected. You'll always want to back up the disk, and sometimes want to modify the program to suit your needs.
3. The program should come with clearly written, thorough documentation that includes an index and table of contents.
4. Bugs and crashes have no place in commercial software! Imagine you're working on a pencil drawing when suddenly the graphite slides off the paper, leaving you with a blank page. Impossible? Well, with a computer image it isn't. You can be working on a graphic, the computer beeps, your picture disappears, and you get a cryptic error message on the screen. Good software should trap errors, and, if there's a problem, allow the user to rescue the picture.
5. The program should allow you to cancel, or undo your last action. (The ESCAPE key is often used for this function.) If you inadvertently press the wrong key, you should not have to sit by helplessly as your picture disappears in a flash.
6. You shouldn't have to be a contortionist to use a program. There are programs that ask you to hold two keys down with one hand while maneuvering an input device with the other hand. Ouch!
7. Routines should be usable without licensing fees. Many software publishers require a credit line if you use their programs for commercial purposes—that's fair. Programs that require licensing fees are seldom used by us, since our clients aren't willing to add extra costs to their projects.
8. Support by the publishers of a program is important. Software that indicates "version 1.3" or "version 2.0" tells you that the program has been updated since it was first introduced. Publishers usually improve their programs after receiving feedback from the users. We do, unfortunately, expect "bugs" in new programs, but a program that has been on the market for over a year and is still difficult to use shouldn't be supported.

What's copy protection?

Under normal circumstances, any Apple file that you save to a diskette can be copied. You can give a disk of files to friends, who can then use a file copy program, such as the one that comes on the Apple System Master Disk or on the ProDOS filer, to copy the files from your disk onto theirs.

The advantages of being able to copy files from one disk to another are obvious—you can swap programs with other people, and you can make back-up copies of disks, so that if a disk gets ruined you have another.

The disadvantage of being able to copy disk files is obvious too. What if you don't want anyone else to be able to use your files? There is a way to prevent it. Altering the way files are saved to a disk, by playing with tracks, sectors, and other mysterious workings of the disk operating system, can make a disk uncopyable, or "copy-protected." Most commercial software is copy-protected so that people can't buy a program and "share" it with others.

"Pirate" is the name given to anyone who copies commercial software. This title is applied to those who copy a program for the purpose of distributing it illegally, those who make copies for friends and family, and those who just enjoy the challenge of cracking another "unbreakable" copy-protection scheme.

We feel that software should be unprotected so that we can make back-up copies or customize it by changing data in the program. Software developers understand our position but claim, perhaps justifiably, that they can't afford to leave their software unlocked (another term for unprotected). Still, some publishers have continued to sell unprotected graphics software. Others have compromised by using a protection scheme that allows the purchaser to make one back-up copy of the disk. (This doesn't help us when we need to alter the program.)

Artists who are dependent upon commercial software as professional "tools" are the real victims in this conflict. It's incumbent upon us to encourage publishers of unprotected software by honoring their trust. Under no circumstances is it excusable to distribute or receive copies of commercial software.

What's public domain software?

Public domain software refers to programs that are available for free or for the cost of duplicating the program on a diskette. Public domain software can be obtained from computer users' groups, electronic bulletin boards, and sometimes in libraries.

These programs may or may not be copyrighted. Some programmers write software and then offer it, at no cost, to anyone who wants it. Sometimes a manufacturer of software may put a program into the public domain because it's old and no longer sells. Most public domain programs are written by computer hobbyists who wrote the program for the sheer fun or the challenge of writing it.

Some public domain software is virtually worthless as it's riddled with bugs. There's seldom any documentation available for this software, and it always comes "as is." Sometimes, however, a public domain program can be a very useful one.

The main value of public domain software is to people who want to learn how to program, because you can list, copy, and change the programs. Many people have learned programming by getting these programs, figuring out how they do what they do, and then enhancing them.

Are there special programs I should use?

Graphics software for the Apple computer ranges from simple, inexpensive drawing and painting programs to complex, costly Computer Assisted Design programs. With the right software, you can draw perfect geometric shapes, draw freehand, paint, move parts of drawings around (cut and paste), add text in hundreds of different fonts, animate images, change colors, enlarge images, reduce images, flip images . . . the list goes on and on.

The incredible part of all this is, you don't need to know how to program a computer to do these things. You merely hit a key, or a button on a joystick or other input device. You're the artist, the screen is your canvas, and with the Apple computer the possibilities are endless.

Within the confines of one book, we couldn't possibly discuss all the graphics software that's on the market. We've selected the packages that we felt were priced within the range of the average Apple microcomputer user and that are most frequently used by us and other computer graphics artists we've spoken to.

A more complete list of graphics software can be found in the Appendix. Be aware that new software is always coming out—keep up to date by visiting Apple computer dealers and by reading magazines that specialize in the Apple computer. In the Appendix you'll find a list of the current periodicals.

Apple Mechanic

by Bert Kersey **Beagle Bros. Micro Software Inc.**

Apple Mechanic is a multiple-utility package. First and foremost it's a shapemaker utility, and that's why we've included it here. But it also contains a program, Byte Zap, that lets you change data on your diskettes. (See How can I remove files from my disk? p. 63) Byte utilities are generally too complicated for the beginner, but you should be able to use this one.

Beagle Bros. is known for high-quality, well documented, unprotected programs. With every diskette you get charts and a manual of tips on using your computer.

Apple Mechanic has a Shape Editor, Shape Analyzer, and Font Editor. Shape tables are limited to 12 shapes, so this isn't the software to use for a large shape table. Shapes created with this program are compiled differently from standard Applesoft shapes, so you may not be able to use Apple Mechanic shapes in shape tables created with other software (and vice versa).

You use the keyboard to plot shapes on a grid that's three times the size of the actual shape. In a corner of the screen is a smaller box that mirrors your plots. The magnified grid assures accuracy while you work, and the smaller box lets you see the shape as it will appear when you use it.

The Shape Editor lets you "preplot" shapes—you plot out your shape but your moves aren't recorded. Then, once you're satisfied with the way the shape looks, you retrace the moves and the shape goes into your shape table. You can also "imprint" a shape—use any shape in your table as a guide in another grid. This is helpful when you're doing a series of sequential shapes. For example, if you're creating a walking figure, you may want only the position of the legs to change from shape to shape. Once you've plotted shape 1, you can imprint it onto the grid for shape 2 and trace over everything but the legs. Add the new leg position and shape 2 is finished—work time is cut in half.

The Font Editor program is different from other hi-res character programs, such as Higher Text, so you can't use these fonts with standard hi-res text generators. The fonts you create with this utility are actually shapes, not hi-res fonts. These fonts can be proportionally spaced, drawn and Xdrawn, positioned anywhere on the screen, rotated, scaled, and so on. Apple Mechanic includes routines that let you use these fonts in your own BASIC programs.

Beagle Graphics

by Mark Simonsen **Beagle Bros. Micro Software Inc.**

Beagle Graphics is a double-resolution graphics package. It's well documented, un-protected, and it includes handy reference charts.

The program comes on a double-sided disk: one side is a DOS 3.3 version and the other side is a ProDOS version—both versions work identically. Pictures created with any of the utilities can be converted from 3.3 to ProDOS and back using the CONVERT program on the ProDos user's disk.

Beagle Graphics offers an enhanced Applesoft which uses the ampersand hook to give you double-hi-res commands (DHGR), and double-lo-res commands (DGR). In addition to DHGR and DGR, there's a font editor, a slide show and many other utilities, including one which lets you easily convert standard lo-res and hi-res pictures to double-res pictures. There are also routines to convert hi-res and lo-res programs to double-hi-res and double-lo-res programs.

Beagle Graphics includes DOUBLE.SCRUNCH for picture packing and unpacking—a real boon when using disk-hogging double-res pictures.

FONT.EDITOR lets you create double hi-res character sets that can be saved to disk and used from DOUBLE.PLOT's Text mode, or from within your own programs.

SLIDE.SHOW creates presentations of your double-hi-res pictures. You can use the pictures as you created them, or use DOUBLE.SCRUNCH to pack them first.

Beagle Graphics includes DOUBLE.PLOT, a double-hi-res drawing and painting program. DOUBLE.PLOT is not as easy to use, nor does it have as many options as Dazzle Draw, or The Complete Graphics System which are described later on in this section.

As of this writing Beagle Graphics is the most comprehensive double-resolution graphics package available for the Apple computer.

Blazing Paddles

by Michael Darooge **Baudville**

Blazing Paddles is a high-resolution drawing and painting program. Like all of Baudville's software, the program is menu-driven and exceptionally easy to use. The disk is copy protected.

This program contains all the options we've come to expect from high-quality graphics software including lines, boxes, circles, text, shapes, zoom, spray paint, fill, and brushes. It offers editing options such as cut and paste and "undo," which allows you to cancel your last action.

One outstanding feature of Blazing Paddles is the iconic menu that frames the screen. When you move your cursor off your picture and press the input button on whatever device you're using, the menu appears. The menu frames the screen so that while you're selecting your tool, you can still see most of the graphic you're creating. This is preferable to the usual procedure where, when you go to the menu, you lose sight of your picture. Sometimes you forget what you went to the menu for!

As soon as you make your selection by moving the cursor over the chosen icon, the menu disappears and you're returned to the full-screen image. The menu doesn't stay on the screen and obstruct a part of the graphic as some other graphics program menus do.

Blazing Paddles has a fascinating Zoom mode that makes pixel editing much less tedious than it often is. Color mixing options are good, and an additional bonus is that you can load in shape tables and use the shapes in your graphics. Included also is a Printer Graphics Dump.

While it accepts input from joysticks and the touch tablet, Blazing Paddles works best with the Apple Mouse II and the Apple graphics tablet.

The Complete Graphics System

by Mark Pelczarski, David Lubar, and David Shapiro **Penguin Software**

The Complete Graphics System, commonly referred to as CGS, contains utilities for drawing, painting, font editing and generation, shape creation, image manipulation, and 3-D graphics. The disk has standard hi-res versions of the programs on one side and double-hi-res versions on the other side.

Penguin graphics software isn't copy-protected and the company offers full user-support, including free license to use their graphics routines, as long as they're given a credit line when the product is used commercially. Graphics software from Penguin is constantly updated, and the company replaces old versions with new ones, for a nominal fee.

Penguin's documentation gives you all the information needed to use the graphics created with their software in your own programs. You're encouraged to take graphics routines from their disk and copy them onto your own disk.

CGS accepts input from any graphics device. This program excels as a painting utility. You can choose from 96 different "brushes," ranging from a one pixel brush for fine detail, thick brushes for boldly laying on of color, to brushes that offer unusual airbrush effects. The palettes in both the hi-res and double-hi-res versions offer an extraordinary selection of colors and patterns. Pictures can be magnified 2, 4, and 8 times for editing. A "Tricks" portion of the program allows you to create mirror images of your picture, move parts of your image to different areas of the same picture or onto another picture, shrink a picture, and flip the colors in your picture.

The program comes with a large and small font, and you can use the font editor to create your own characters. A compatible disk of alternate fonts, called Additional Type Sets, can be purchased.

The utilities in this program are easy to use, with the exception of the 3-D graphics editor, which requires studying.

Dazzle Draw

by David Snider **Broderbund Software Inc.**

Dazzle Draw is a powerful double-high-resolution paint program. With this program and an Apple //c, or an Apple //e with an extended 80-column card, you can draw and paint on the hi-res screen with all the benefits of double-hi-res graphics—16 colors and a horizontal resolution of 560.

Dazzle Draw's copy protection scheme allows you to make one copy of the program disk. Dazzle Draw accepts input from just about any graphics device, but the Apple graphics tablet and Mouse II work best.

This program excels as a drawing and painting utility. The pull-down menus offer numerous options, including shapes (solid and outlined ellipses and rectangles), lines, brushes, fills (including filling over patterns), color mixes, patterns, text, grid, mirrors, zoom edit, cut and paste, and color exchange. The color exchange option allows you to select any color in your graphic and swap or replace it with another color. In addition there's an Undo option which lets you undo the last move you made. With the pattern editor you can create your own color mixes and save them to disk for future use.

This comprehensive package also includes a printer dump utility and a slideshow program. The slideshow is limited to six slides per show since that's the maximum number of double-hi-res graphics that will fit on the disk. Unfortunately, the program doesn't have a picture packing utility.

The pull-down menus make the options in this program easy to use, but there's a tradeoff. The menus, always visible, span the top and bottom of the screen, blocking off part of the display. There's a scroll bar to scroll the image as you work, as well as a view full screen option, but you'll find it's a serious disadvantage to be unable to see the entire image while you're working.

The Designer's Toolkit

by Eclectic Electric **Apple Computer Inc.**

The Designer's Toolkit is a high-resolution drawing and painting package that can be used only with the Apple graphics tablet. The software comes with a mylar overlay menu that's taped to the tablet and functions like the overlay that comes with Graphics Tablet Software. (See Apple Graphics Tablet with Graphics Tablet Software, p. 247.)

We hesitated to include this package in this section as it's been impossible to buy at the time of this writing. We decided we should, because it's the software that we use most often, and if you're aware of its existence you might, somehow, find it.

The Designer's Toolkit has five programs:

1. Text Generator is a hi-res character generator with 15 fonts.
2. Color Definition lets you create colors (160 are possible), save them to disk, and use them in the painting program.
3. Brush Definition allows you to define (create) your own brushes. Sets of brushes can be saved to disk.
4. Review Pictures is a slideshow utility.
5. High Resolution Tablet is the drawing-painting program.

Besides the usual graphics options such as draw, line, box, and so on, High Resolution Tablet offers many unique features:

With Invert you can invert the colors of all or a part of your image, and with Reflect you can make a horizontal or vertical mirror-image of all or a part of your graphic. Separate lets you create color separations and Magnify enlarges your image for pixel by pixel editing. Slide lets you scroll the entire screen, and Window designates an area of the screen in which all options will be active. Once you designate a window, most options will only affect the part of the image that's within the window.

The most significant feature of this paint program is that you can work on both hi-res screens, interactively. Pressing the stylus on the appropriate menu box takes you to and from either screen where you can use all of the graphics options. Using any of three possible Merge options, you can combine images from one page with the other.

The Designer's Toolkit is a program we highly recommend for use with the Apple Graphics Tablet. Good luck finding it!

DOS Toolkit

Apple Computer Inc.

DOS toolkit is a mulifaceted package. The original release contained a 6502 Assembler and simple text editor (EDASM), an Applesoft program editing tool (APA), and a high-resolution character generator and font editor (HRCG and Animatrix). In late 1984, Apple replaced APA with the Boston Window, another Applesoft editing tool. Currently, DOS Toolkit supports only DOS 3.3; we expect this to change.

While the Applesoft editing utilities are highly recommended, DOS Toolkit earns its place as a graphic tool by virtue of its hi-res character generator and font editor.

HRCG, unlike Higher Text, doesn't support colored or large fonts. Nonetheless, there are two important reasons to use HRCG. First, most Apple software developers have already licensed HRCG for distribution on their commercial software. (See Can I use other's routines in my work?, p. 266.) Second, HRCG is the only hi-res character generator designed to do character/block animation.

When HRCG is active, all text is displayed on the hi-res screen. Apple used control codes to command HRCG. For example, CONTROL P will clear the screen and CONTROL A will switch to an alternate character set. (You can have as many fonts as fit into memory.) CONTROL B and CONTROL D delimit a character block. All characters between these codes are displayed as a block, as if the characters were glued together. The effect is not unlike the Text Animate fish. (See Graphic Bonus:Text Animate, p. 181.) The HRCG control codes allow for nearly every type of character/block animation you might require, including an XOR mode for background conservation. A game, RIBBIT, included on the disk, demonstrates HRCG's capabilities.

Without a good editor, HRCG would be useless. Animatrix is the HRCG editor. As the name implies, it was designed with animation font editing in mind. Although an HRCG font is 7 by 8 pixels, the edit grid is large: 7 by 5 text characters (or 49 by 40 pixels). While you're editing in the magnified edit area, a small "graphic" window shows the figure in it's actual size. The only major deficit in the editor is color control; you must skip pixels to put color in a character. You can control whether a byte will have it's high bit set for color set two (orange, blue, white 2) or for color set one (green, violet, white 1). This program requires paddles—joysticks and touch tablets don't work well.

Edu-Paint

by Computer Advanced Ideas **San Juan Unified School District**

Edu-Paint has a precarious position in this section as we've been unable to determine its availability. The program was developed in 1981 for a school district and although many people have it, it doesn't appear on any commercial software list. We've been led to believe that it's a public-domain program but this hasn't been confirmed. The software is not copy-protected.

Edu-Paint is a drawing and painting utility that offers many options. The program allows you to draw lines, boxes, circles, filled boxes and filled circles. There's a fill option, but there are no paintbrushes.

Although the program accepts drawing input from most graphic input devices, all option commands must be entered from the keyboard. Fortunately, the commands require only one keypress, such as F for Fill. Two options that make the package worth having are the Image and Color Menus.

The Image Menu lets you define an area of your graphic and save it as an "image." This image can be cut, copied, moved, and saved to disk for use on another graphic. The Cut and Paste option is not unique anymore, but remember that this program offered it in 1981, before any other graphics program did.

Edu-Paint's Color Menu is unique. According to the documentation, you can find or develop over four billion colors and color patterns. Although we haven't put this statement to the test, we haven't yet found a limit to the number of colors and patterns available.

Colors can be "mixed" and saved as pattern sets. You can define hundreds of sets, such as "Blues," "Greens," "Stripes" etc., and save them to disk. When you need a particular set, load it from disk back into Edu-Paint.

One warning if you use Edu-Paint: The Clear screen command doesn't ask for verification. One touch of the C key and your picture is gone!

E-Z Draw

by Nasir Gebelli and Jerry Jewell **Sirius Software**

E-Z Draw, a hi-res drawing utility, is one of the older graphics packages; it may be difficult to find in local stores.

This program accepts only keyboard input, which makes it tedious but precise. You control the crosshair cursor with the four diamond keys, I, J, K, M or ↑, ←, →, ↓, on the //c and //e. The cursor can be moved in increments of one pixel to nine pixels at a time. With E-Z Draw you can plot dots, lines, parallelograms, rectangles, triangles, circles, and ellipses, outlined or filled, in any one of the standard Apple hi-res colors. E-Z Draw generates circles and ellipses with a better aspect ratio than most graphics software programs. The circles are round, not elongated or squat. By hitting a key, to stop the circle from plotting, you can create arcs of any size. You also have the option to set your own scale from X,Y to XO,YO (default is the normal hi-res page scale, X O>279 and Y O>191).

E-Z Draw allows you to manipulate "images," selected parts of the screen, as well as the entire screen. You can move, duplicate, expand, compress, invert, flip, rotate, save, and load images. Images frequently get distorted when you select some of these options, so, as with any graphic generated on the computer, constantly SAVE your picture or image as you go along.

E-Z Draw uses the character generator from Higher Text. You can select one of 18 fonts from the disk and put text on your graphic in a number of colors and sizes. There are many options in the character generator, but it'll take concentration and practice before you'll be able to take advantage of them. A font editor is not included.

E-Z Draw doesn't have the enhancements of newer graphics software packages, such as rubber-banding, color fill, color-mixing, and alternate input-devices.

Fontrix

by Steve Boker **Data Transforms, Inc.**

Fontrix is a high-resolution character generator, and more. The program is a typesetting, drawing, and printing tool that provides you with a large selection of traditional and decorative fonts and a one-of-a-kind graphics utility. With this software you can create full-page illustrated documents, such as flyers and newsletters, on the Apple computer.

Using the Font Editor, you can design proportional or nonproportional fonts with characters as large as 32 × 32 pixels—this allows for finely detailed character sets. The supplemental Fontpaks, published by Data Transforms, are proof positive of the high-quality character sets, letters, symbols, and images that are possible with this utility. We use Fontrix extensively for title screens and other graphics that require text.

In addition to standard 8K images, the Graphic Writer lets you create "extended screen" graphics. When you're working with the 26 available options, you aren't limited to an image the size of the Apple screen. You can open a Graffile of your own size specifications and produce an image many screens wide and many screens tall. As you work on the Graffile, when the cursor reaches the edge of the screen, the screen scrolls horizontally or vertically. Using the various options, you can input text, change fonts, set windows that define the working area, load in graphics created with other programs, and/or create new graphics using the graphic writer with just about any input device.

When working on a Graffile it's easy to get lost scrolling from screen to screen, so it's imperative that you plan your layout in advance. It's also helpful to scroll through the blank Graffile when you begin, putting small marks as guidelines at the screen edges.

A Graffile wouldn't be very useful if you had to boot Fontrix and scroll through it when you wanted to show it. In actuality, the main function of this program is not to create screen graphics but printed hardcopy. When you print your Graffile, unique routines print the entire Graffile as one full-page document. The printer drivers work with just about every printer and printer interface card, black and white or color.

GraFORTH

by Paul Lutus **Insoft**

Like Logo, GraFORTH is a programming language. Unlike Logo, however, GraFORTH is intended for professional graphics applications and features a rather complete set of graphic functions: standard coordinate graphics, turtle graphics, three-dimensional graphics, and character animation. It also supports music. There's a penalty to pay, however, for GraFORTH only supports integer—whole number—variables and is significantly more cryptic than Logo or BASIC.

GraFORTH is derived from Forth, a programming language invented in the 1960s to control telescope movement. Forth was intended to be compact and fast, without the verbosity of other, more sophisticated languages. GraFORTH is only distantly related to the older Forths.

Forth and GraFORTH work through a first-in, first-out stack. A stack is like a plate dispenser in a cafeteria. Plates are pushed into a spring-loaded contraption and the most recently added plate (the one on top) is the first to be removed. To add two numbers, for example, one types "2 5 + .". GraFORTH puts 2 on the stack, then puts 5 on the stack, then added the top two numbers on the stack, and places the result on top of the stack. The period pulls (removes) the top number from the stack and prints it. Forth admirers find this way of working simpler than working with BASIC.

The benefits of GraFORTH, however, may outweigh the language difficulties. GraFORTH is much faster than BASIC, yet it's much more understandable than assembly language. Secondly, there's a rich collection of graphics commands as well as the ability to create your own. GraFORTH is extendable. By combining GraFORTH commands, you can define new ones—something like, but much more powerful than, BASIC's subroutines. Thirdly, GraFORTH uses the hi-res display for text display. You can mix text with graphics and you have the option of having several character sets in memory at once. By designing the character sets in memory at once. By designing the character sets as graphic blocks, GraFORTH is a natural for animation. Finally, GraFORTH has built-in editors to ease creation of programs, animations, and 3-D graphics. The only thing missing is a paint program, but you can use the images from other paint programs with GraFORTH programs.

The Graphics Department

Sensible Software, Inc.

The Graphics Department is a business and presentation graphics utility.

While most Apple graphics artists don't need this type of software, some of you may find that you're called upon to create many charts and graphs, or even a complete presentation. (See What are presentation graphics? p. 171.) We felt that at least one presentation graphics utility should be included in our software overview and The Graphics Department, which isn't copy-protected, is the one we recommend.

This program distinguishes itself from other presentation graphics utilities in its ease of use and extensive capabilities. The software combines plotting, chart generation, lettering, graphics editing and a slide projector, all in one package.

The Graphics Department offers many special options:

1. The chart generator creates pie, bar, scatter, and line graphs from data that you key in or from DIF files. (DIF files are special files created by programs like VisiCalc.)
2. You can choose lettering from twenty character fonts. Fonts can be headlined and shadowed, and can be input in a choice of sizes, colors, and directions.
3. Images can be cut and pasted, overlayed, merged, flipped, shrunk, inverted, and so on.
4. Standard 8K pictures can be created with the system, or you can use it to edit pictures created with any other standard Apple graphics utility.
5. The Slide Projector module gives you control of up to 32 high-resolution pictures for display on your CRT. These pictures can be subtitled and shown at varying speeds. Your slideshow can be shown interactively or as a self-running demo.

The Graphics Department is easy to use. All the available commands for each module are shown on the screen and each command is selected with one keystroke. The documentation is comprehensive.

The Graphics Magician

by Mark Pelczarski **Penguin Software**

The Graphics Magician is an outstanding package, with utilities for drawing, painting, and animation. The software isn't copy-protected.

The drawing and painting utility, called The Picture Painter, is unique. The program records your moves (e.g. plot brush at 10,20) in a text file. When you save your picture, it's the moves that are saved, not the final image. When you display your picture you see it redrawn exactly as you drew it—like a TV instant replay. Working on "sequential pictures," as they're termed, is tedious. Each line and brushstroke must be recorded, one at a time. There's no rapid freehand sketching here, but there are advantages to this system.

The text-file information, like a short BASIC program, takes up considerably less room than a graphic image, so you can fit hundreds of images on a single disk. This is the technique used in many commercial software programs, especially illustrated adventure games.

Your moves are saved and replayed, so you can animate your images. If, for example, you draw an open eye, cover it with a closed eye, redraw it open, close it, and so on, when the sequential picture is displayed the eye seems to blink.

The second part of The Graphics Magician is a popular animation system that uses shifted-shapes. You begin in the shape editor where you create seven variations (shifts) of one shape—the program compiles them. Next, with the path editor, you plot a path for the shape. With shape and path complete, you create an animation file with the animation editor. BASIC and machine language routines for using the animation files are provided on the disk.

The quality of the animations is attested to by the fact that many commercial programs (especially educational software) contain the following credit line on the packaging: "Graphics created with The Graphics Magician by Penguin Software."

A double-hi-res version of the picture painter utility is available for the Apple //c and //e with extended 80-column card. It's called The Picture Painter. Sequential pictures made with the standard Picture Painter are easily converted to double-hi-res sequential pictures.

The Graphic Solution

Accent Software

The Graphic Solution, also known as TGS, is a powerful animation utility. The program is designed to let you create an animation in a manner similar to working on movie film.

You create bit-map shapes using keyboard commands, and create the animation frame by frame. Text can be included in the frames. A frame series is saved as a sequence, and these sequences are shown via a "projector." The sequences and the projector can be saved on your own disks and accessed with a simple BASIC program.

You draw your shapes with any hi-res graphics utility, bring the screen into TGS, and then capture them from the screen. If you prefer, you can create shapes within the program, which uses a low-resolution shape editor and then converts the shapes to hi-res. In the shape editor you can edit a shape in a number of ways: expand it, contract it, scroll it, create a mirror-image of it, and so on, all with a few keypresses.

The animation creation process is as follows: Put a shape in the window, shoot the frame, move the shape a bit, shoot another frame, and so on. Then you go into a SHOW mode, press a key, and see your film.

There are a number of extra features in this package, including an option to print the screens as you're working on them simply by pressing a few keys. You can also save a series of steps, for example "move left, shoot a frame, move up, shoot a frame. . ." as a macro. When you press the designated key, all the steps are automatically carried out. This saves a lot of time and tedium.

The Graphic Solution comes in versions that work with the Koala light pen and the Koala touch tablet. This is a program that requires time and patience to master, but when you do master it, you will be able to create professional quality animations on the Apple computer.

Accent software requires a licensing fee for commercial use of the TGS utilities.

Higher Text Plus

by D. and R. Aldrich **Apple PugetSound Program Library Exchange**

Higher Text Plus is a high-resolution character generator. It's one of the software "classics"—introduced in 1980 and still used extensively. The program is not copy protected.

Higher Text Plus allows you to produce text on either hi-res graphics screen. The disk includes 13 upper- and lower-case ASCII character sets. Using this program, you can print on the hi-res screens using normal Applesoft BASIC commands such as HTAB and VTAB. There are two utilities:

The character generator interprets the BASIC program lines, converts them to instructions understandable by Apple's hi-res graphics routines, and places the characters at the desired locations on the screen. Characters can be printed in several sizes and shapes. This is one of the few character generators that lets you display text in color.

The character editors let you edit or redefine characters. There's one editor for small fonts and one for large fonts. For the professional artist, this would be the program to use when a client commissions an original font.

Higher Text Plus is probably more useful to programmers than to the average graphic artist. If your only desire is to add a few words to an image, you can easily use a paint program that comes with a text utility. If, however, you're working on a program or a resume disk that requires extensive text on the hi-res screen, you should consider a hi-res character generator.

Higher Fonts are diskettes that contain a variety of fonts for use with Higher Text Plus. Each fonts diskette comes with a utility called Font Viewer, which allows you to preview the fonts on the disk.

Apple Logo

Apple Computer Inc.

Logo was developed at the Massachusetts Institute of Technology specifically to teach children about computers and math. Today it's one of the most sophisticated (but not complicated) interactive computer languages.

When Logo is loaded into the computer, it replaces BASIC. Graphics are essential to Logo, and an important part of learning to program in Logo is learning to draw with the tools of the language. No other general-purpose computer language has so thoroughly embraced graphics.

In Logo, the drawing tool is called a turtle. The turtle, a tiny triangle on the screen, represents a robot holding a pen. The turtle is ordered about the screen with commands such as FORWARD 10 and LEFT 90. (Translation: "Walk forward 10 steps or pixels." and "Turn 90° to the left.") As the turtle walks it draws—unless you direct it to hold its pen up.

Turtle graphics, then, are relative—the turtle draws from wherever it's standing. If you write a series of instructions that draw a star, the turtle will draw a star starting at its current location on the screen. A star-filled sky requires only that you move the turtle between drawing each star. A few simple Logo subroutines (called procedures) can create complex images. Pictures created with Logo are standard Apple pictures.

You can also direct the turtle to go to a specific screen coordinate, much like BASIC graphics programming. Other Logo commands allow you to change the pen or background color, ask the turtle where it is or if its pen is up or down, and so on.

Logo also includes a full range of other capabilities for input from the keyboard, joysticks, and so on. It includes loops and conditional branching, everything a programmer needs to write complex programs. It's a **real** computer language.

There are many versions of Logo, each differing slightly in their capabilities. Terrapin Logo is best known for it's openness, the ability to interface nonstandard devices and equipment to the language. Apple Logo and Apple Logo II have more sophisticated file-handling capabilities as well as support from Apple Inc. Both variants of Logo, however, are good choices to learn about programming and learn about turtle graphics.

Pixit

by Michael Darooge **Baudville**

Pixit is a high-resolution shape utility program. With it, using keyboard input, you can create standard Applesoft shapes and shape tables and then use the shapes to create a picture. The disk isn't copy-protected.

Pixit has a Picture Editor for creating hi-res images using shapes, a Shape Editor for designing and editing shapes, and a Shape Table Editor which lets you create and edit shape tables.

The program also includes a Shape Sorceror for analyzing shapes, a printer dump that gives hardcopy of the Picture Editor screens, and shape and font "libraries" (ready-made shape tables).

The Picture Editor lets you load in shape tables and plot the shapes in different scales, colors, and rotations. This is supposed to be a paintlike utility for creating graphics from shapes, but we found the available options too limited to use for professional picture creation. We have, however, found a very practical use for this editor. After we've created shapes (with this or any other shape program), we use the Pixit Picture Editor to make reference printouts of our shape tables. We load in a shape table, plot each shape, use the text option to label and number them, and then use the print option to dump the screen to the printer. Try it. When you've created lots of shape tables, you'll appreciate having these reference charts.

Pixit's Shape Creator allows you to create a shape in any scale you wish, with or without a grid for alignment. It also lets you backstep and edit your shape.

The Shape Table Editor is the best we've used. With this editor you can load shapes and shape tables made with any graphics utility and use them to build new shape tables. Working with two shape Tables in memory—the source table and the new table—you can add shapes, delete shapes, and insert shapes just by hitting a few keys. We use lots of shapes, so we've created "master" shape tables. Then, when we need a special table, for example, five animal shapes, we find the master tables that have animal shapes (and also have other shapes). We load these master shape table into Pixit's table editor and copy the animal shapes we need from them into a new shape table.

The Printographer

By Stephen Billard **Roger Wagner Publishing, Inc.**

The Printographer is a graphics dump utility. With it, you can dump standard high-resolution screens to a printer.

We've found the quality of the black and white printouts made with Printographer superior to most graphics dump software.

Printographer, which isn't copy-protected, works with an impressive variety of printers and printer interface cards. Color printing with color printers is supported. After you boot the program and load a graphic into the computer, you can "dump" or print the graphic in a variety of ways.

You can crop the image horizontally and/or vertically, "mask" the image so that it's diamond or oval shaped, and/or add text to pictures—horizontally or vertically, normal or inversed, upper-or lower-case. The program supplies some fonts, and it also allows you to load in and use alternate character sets from other programs like the DOS Tool Kit and Higher Text.

If you like the way your picture looks after you've made some of these changes, such as shaping it like a diamond, the new version can be saved to disk. Printographer also has a compression or picture-packing utility. Finally, you can print graphics normally (black dots print black) or inversed (white dots print black) as well as magnified up to 99 times. Pictures can be positioned anywhere you want horizontally on the paper.

The documentation comes with instructions on how to use the Printographer's utilities to print graphics from within your own BASIC programs.

subLOGIC Three-D Graphics System

by Bruce Artwick **SubLOGIC Corp.**

Three-dimensional graphics are the least developed area of microcomputer graphics; there must be ten paint programs for every 3-D package. (See What are three-dimensional graphics? p. 174.) The subLOGIC Three-D system is one of the best and most flexible systems available for any microcomputer.

Three-D systems must include an editor to allow you to enter data and a display program to allow you to see your objects. You should be able to display the graphics under control of your own program. The subLOGIC system meets these qualifications.

The subLOGIC editor is well-thought-out. Editing is visual—a flashing cursor is moved through the object. This is important as it allows modification of an object for aesthetic reasons, for the vectors that describe a 3-D object may be correct, but they may not yield a visually pleasing object. Corrections and alterations to any point within the object are made by moving the point to the desired new location. Cursor movement is keyboard controlled. If you need absolute mathematical precision, you can create a numeric file and read it into the subLOGIC system.

The subLOGIC system supports multiple objects within a 3-D creation. These objects may be stationary—bound to the world coordinate system—or independent—moving within their own coordinate system independent of the coordinate system of other objects on the screen. Unique to the subLOGIC system is a motion editor that allows you to zoom, pan, and move through the 3-D world you've created in the object editor. These movements are saved and can be played back under control of your own program. A simpler 3-D playback utility, the slide show, can also be used to generate 3-D animations that run independent of the system.

Text can be added to your 3-D creation as an object. The text, then, is incorporated into the animation and is subject to whatever manipulations you may impose. The characters in the character set are not very well designed, but the manipulations you can perform on them may override the design problems.

The subLOGIC Three-D System is probably the most complicated graphics utility described in this book. The system is driven by single key commands; until you're comfortable with them you'll find yourself paging through the extensive user's manual.

Take 1

by Maurizio Barbatano and Michael Darooge **Baudville**

Take 1 is an animation utility. With this program, using keyboard input you can create computer "movies." No programming skills are required. The disk is copy-protected.

Take 1 is relatively easy to use. We qualify "easy," because no animation software can do all the work for you. For good animation it's essential that you begin with a well-thought-out storyboard—this has to be done before you even boot the software. Once you've storyboarded the action and designed your characters on paper, then you're ready to begin using an animation utility.

The program is menu-driven and has no complicated commands to learn. Once you've read the small manual and spent an hour or two experimenting, you'll be ready to go.

Take 1 consists of Pictures and Backgrounds, Actors and Actions, Scene Editor, Movie Editor, Movie Projector, and Disk Utilities.

Create your background scene and characters by drawing them on the hi-res screen with any hi-res paint utility. Then load the screen of characters into Take 1's Actors and Actions. Here the characters can be edited and "snapshots" taken, so they become "actors." The background scene gets loaded in and the actors are "cast" in the Scene Editor. The actual "movie" is made in the Movie Editor, where one or more scenes are spliced together. One movie can be several minutes long, depending upon the speed of the animation and the disk space.

Each utility has options that allow you to get wonderful animation effects, including sound, variable frame speed, canned actions, and "sprite" snapshots (the actors can be created with transparent areas in them so that, for example, if a car is used, the background would be seen through the car's windows).

Movies made with Take 1 can't be shown without the program's projector utility. Currently there's a licensing fee for its use. This isn't a problem if you want to create animation for personal use, but if you intend to use your movies for commercial applications, you'll have to negotiate with Baudville.

Transitions

by Andre Schklowsky **Penguin Software**

Transitions is a high-resolution slide show utility used to create graphics presentations. The disk is not copy-protected.

This program lets you choose from any of 44 possible screen transitions, or wipes, between pictures, and you can vary the speed at which the transition occurs. A Wipe Sampler is provided. Some of the transitions can be used to create impressive effects as the new picture appears on the screen and wipes out the previous picture in sections.

Transitions lets you pack standard 33 or 34 sector hi-res pictures. The program can access up to eight disk drives per presentation, and your presentation can be manually controlled or run automatically. The back side of the disk contains a Master copy of an Automatic Packed Picture Slide Show which allows you to create bootable presentation disks.

Transitions contains some unique utilities:

Hi-Res Catalog lets you view your pictures in miniature—1/9th of their original size—on a 3×3 grid with, or without, titles.

Hi-Res Sorter lets you view standard format pictures at 1/16th of their original size on a 4×4 grid so you can rearrange their order in the catalog. You can then save the new catalog to disk. Sequencer lets you arrange the order of packed pictures.

This is a powerful program, but it requires patience to use it—you'll have to spend a considerable amount of time designing, testing and creating your slide show. When you're finished though, you should have a quality presentation.

What do I draw and paint with?

The Apple supports a variety of graphics input devices. Some devices plug into the Apple IIplus or //e internal slots, and some plug into the Apple's game port. Game port devices can be used with the entire line of Apple II computers, but peripherals that require a slot can't be connected to a //c unless they've been adapted for connection through a serial port. (See What's a peripheral? p. 11.)

Keyboard input

All Apples sport one input device: the keyboard. The keyboard allows precise movement, excellent for pixel by pixel manipulation. Most software will use the I, J, K, and M keys for cursor movement; most new software recognizes ←, →, ↓, and ↑ on the //e and //c computers. The keyboard isn't a satisfactory tool for freehand drawing and painting—for these you'll want one or more specialized graphics input devices.

Gameport devices

Input devices that plug into the gameport include paddles, joysticks, touch tablets, and other specialized devices.

Interfaced peripherals

Graphics input devices that are connected to the Apple via interface cards include light pens, graphics tablets, mice, and digitizers.

All input devices have their advantages and disadvantages. For creating shapes, for example, we prefer to use the keyboard, but for freehand drawing and painting we prefer the Apple graphics tablet. Both of us, however, have used the keyboard to create effective full screen graphics. One reason for purchasing a particular device is to be able to use the software package you like, since the software dictates what device or devices can be used.

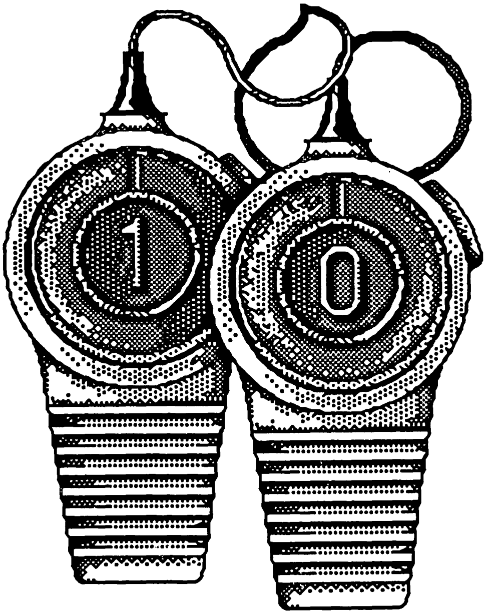
No computer graphics input device totally simulates traditional artist's tools. The computer, however, is not a traditional medium.

What are paddles?

Paddles are analog input devices. The paddle mechanism and the game port circuitry convert a twist of a knob into a computer signal.

Essentially, the paddle is a rheostat—like a dimmer for lights. A small voltage is passed into the paddle from the Apple; the position of the paddle knob determines how much returns to the Apple. This voltage level is then converted to a number by the internal gameport circuitry and programming. Under BASIC, each paddle returns a number from 0 to 255—one byte of information. Assembly language programmers can use special techniques to return a larger range of numbers from the paddle circuitry, but these techniques may not work reliably on all Apples. Apple II, IIplus, and //e computers support four paddle inputs, but the Apple //c supports only two.

Most graphic programs use two paddle inputs. One paddle delivers an X coordinate, while the other delivers a Y coordinate to the program. Thus, if you wanted to draw a horizontal line you would work only with the X-coordinate paddle. If you wanted to draw a vertical line you would use only the Y-coordinate paddle. To draw a diagonal line or draw freehand, you have to manipulate the two paddles. Some programs, such as character set editors, font editors, and others that require predominately unidirectional movement, work best with paddles.



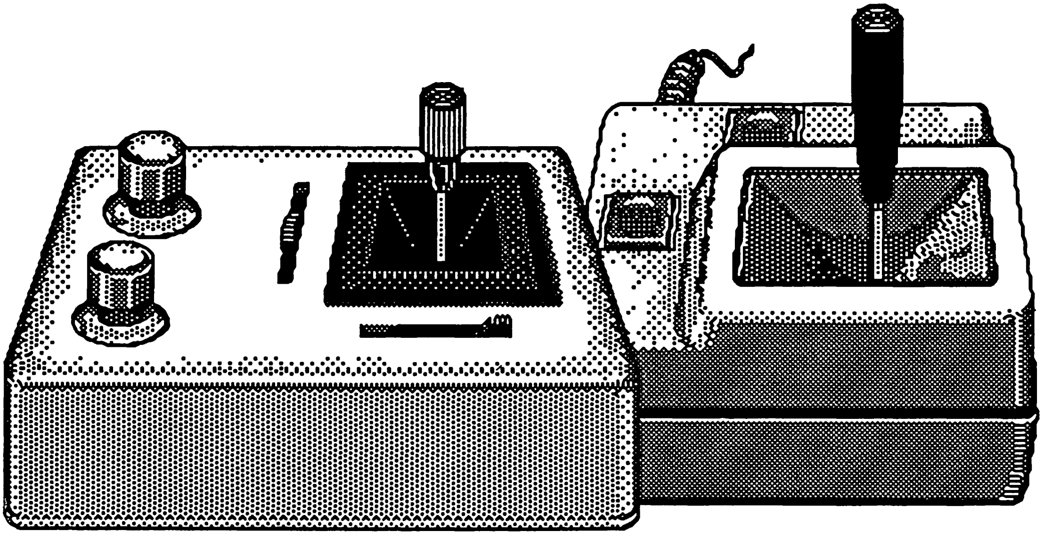
What's a joystick?

A joystick is equivalent to two paddles. If the stick is moved only horizontally or only vertically, only one of the paddle inputs is changed. The stick, however, can move freely in both directions to provide the diagonal movement essential to painting programs. When released, the stick pops back to the center position. This is called self-centering.

In addition to the stick, most joysticks have two buttons (each equivalent to a paddle button). Button input is accepted by software for a variety of functions, most commonly to alternate between drawing or painting and cursor movement.

Moving the joystick results in two numbers, ranging from 0 to 255. One number is used to represent the Y-axis, and the other to represent the X-axis. The joystick range is too large on the Y-axis and too small on the X-axis, so the values are scaled by the program before they're used.

A good joystick will allow you to easily adjust the range of values a movement produces, and will allow you to turn off the self-centering mechanism so that the stick remains in whatever position you put it. Drawing with a joystick is easier than drawing with paddles.



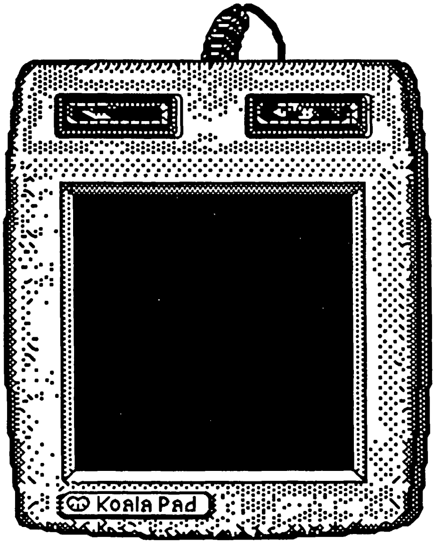
What's a touch tablet?

There are several brands of touch tablets available for the Apple, all of which use slightly different technologies. Unlike the more expensive bit pads or graphic tablets, touch tablets plug into the gameport of the Apple.

The touch tablet has a pressure sensitive surface, usually small in area. When pressure is applied to this surface, from a stylus, edge of a pen cap, or even a finger, the tablet reports two values to the computer via the paddle 0 and paddle 1 inputs on the gameport.

Because touch tablets signal the Apple via standard paddle inputs, they can usually be used with most programs that utilize paddles or joysticks. The touch tablet, however, may not work well if the program is utilizing a special feature of joysticks or paddles. For example, programs which use the self-centering feature of joysticks won't respond well to touch tablet input.

Touch tablets are not as sophisticated, technologically speaking, as graphics tablets and they're not as responsive, but for freehand drawing and painting they're superior to paddles and joysticks.



What's a light pen?

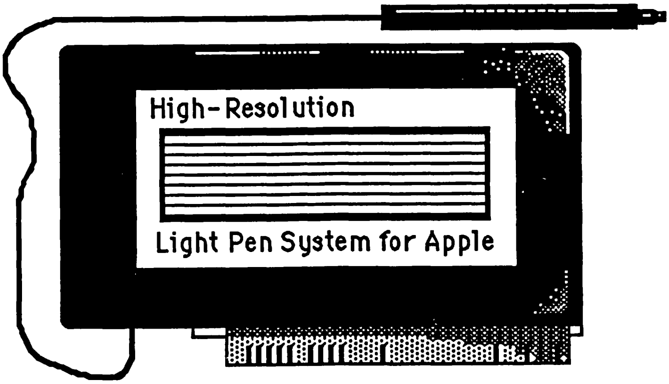
Light pens are perhaps the most interesting of the graphics input technologies. Light pens use the light from the CRT to get input coordinates. A tiny photoelectric cell inside the pen tip generates a voltage when it passes over a lit dot on the CRT. This voltage is passed into the computer. Software and circuitry inside the computer decode the signal into X,Y coordinates. To do this conversion, the light pen has to know which dot was lit when the voltage was generated.

Inexpensive light pens that plug into the computer's gameport require selective parts of the screen to be lit up in blocks. The smallest unit these pens can detect is the size of a text character, so gameport light pens are low-resolution pens.

Light pens that are attached to interface cards use the Apple video circuitry. The Apple is controlling the video display, so signals on the internal Apple bus can be examined to determine where the electron sweep of the CRT was when the signal from the pen was detected. These light pens offer much higher resolution—theoretically the size of a single hi-res pixel.

The idea of drawing directly on the screen is appealing, but there can be problems in using a light pen. After long sessions, arm fatigue sets in. Resolution is another problem—although hi-res light pens can theoretically resolve a single pixel, the size of the pen tip combined with the distance from the screen make it difficult to draw single pixels. Another problem is that there's no generalized Apple light pen interface—unless a program is specially modified, it won't work with a light pen.

Despite these limitations, many artists enjoy the light pen as a graphics input device for the Apple.



What's a graphics tablet?

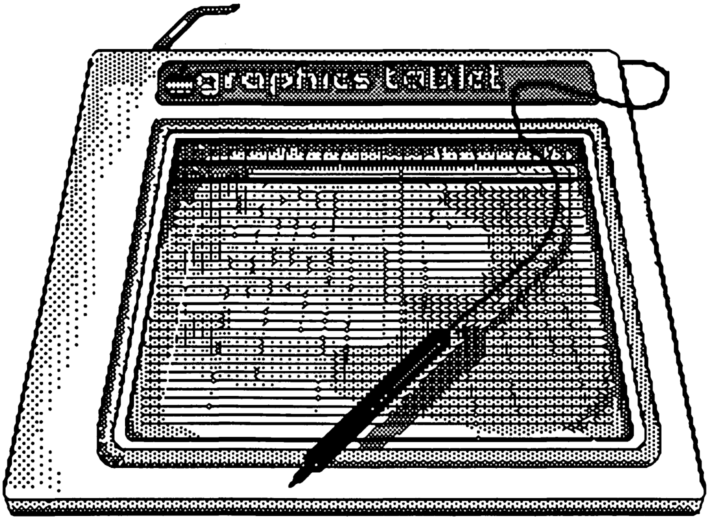
Graphics tablets are also known as bit pads. These devices are large (14″ × 14″ minimum) surfaces with an attached stylus. Don't confuse bit pads with touch tablets; the former costs about $800, whereas the latter usually costs less than $100.

The Apple Graphics Tablet is a magnetic tablet. Under the surface of the pad, there's a grid of wires through which a small amount of current flows. When the stylus is moved over the pad's surface, it's location is sensed on this grid. Software in the Apple translates the signals from the stylus into X and Y coordinates on the graphics screen.

The Apple Tablet has a pen-shaped stylus with a retractable point. The computer can sense whether the point is up or down. The position of the stylus point can be used by software to determine when to draw, when to move the cursor without drawing, what choice to make from a menu, and so on. Drawing on the tablet is like drawing on a piece of paper, except that the image appears on the CRT rather than on the tablet.

Another popular tablet, the Hi-Pad, manufactured by Houston Instruments, attaches to the computer through a standard serial interface rather than a dedicated graphics tablet interface card. Add-ons to the Hi-Pad include a variety of styluses, one mouse-shaped and another with a real pen in its tip.

Other types of graphics tablets are conceptually similar to the magnetic tablets, but they utilize infrared signals, ultrasound, and other technology.



What's a mouse?

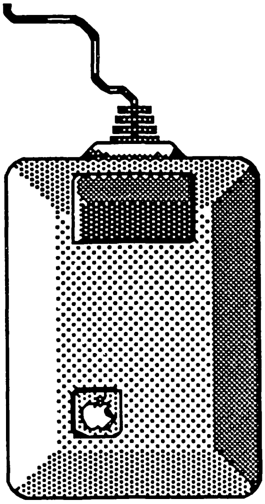
The mouse is one of the newer graphics input devices for the Apple. Currently the most popular mouse is The Apple Mouse II—it's about the size of a box of lime Jello®.

For the Apple IIplus and //e computers, the mouse is packaged with a circuitboard that plugs into a slot; for the Apple //c, which has the circuitry built-in, no interface card is needed and the mouse plugs into the gameport. A small rubber ball is embedded in the belly of the mouse. Rolling the mouse across a surface causes two numbers ranging from 0 to 65535 to be sent to the computer. These values are read and scaled to provide X and Y coordinates for graphics.

The mouse has a button which can be used much the same as joystick and paddle buttons. Unlike joysticks, however, Apple provides an impressive amount of ROM programmer support for the mouse. This support enables very sophisticated mouse activities. We're certain that most future graphics software will include the mouse as a standard input device.

The mouse is a very accurate and responsive input device, but it requires getting accustomed to hand and arm action rather than finger and wrist action when you draw.

It's important to note that the mouse can't be used in place of game controller input devices such as the joystick. Software must be specifically programmed to use it.



What's a digitizer?

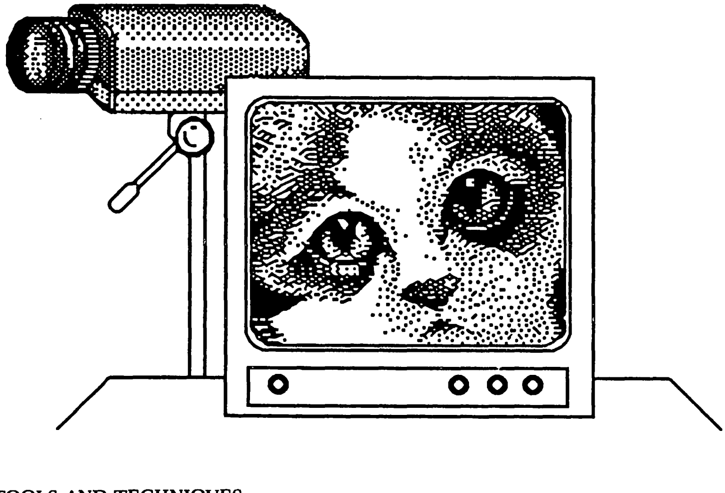
A digitizer is a hardware device that converts our analog, or continuous world into the digital, or discreet world of computers. There are many varieties of digitizers.

Some digitizers attach to plotters or printers while others accept input from a TV camera. Regardless of the source of the signal or the specific mechanism, digitizers allow you to convert a still or live image into a computer graphic. You can actually photograph people, animals, objects, or pictures from books and magazines, and the image appears on the CRT. You can then edit and save the digitized image like any other computer graphic. The possibilities are endless.

Our visual world is analog—our eyes are able to sense a range of hues and tones. A digital device like a computer can't deal with subtle ranges; a pixel is either on or off. Much like halftone printing, the digitizer converts a range of tones into a group of pixels. Where the original photograph or graphic is dark, the digitizer will turn on only a few pixels; where the original is light, the digitizers will turn on lots of pixels. The final image is composed of blacks, whites, and a range of gray tones.

Digitizers come with software. Good software allows you to control the intensity levels being digitized, allowing you to create "high contrast" as well as normal images.

Although digitized images are blacks, white, and gray, when the image is displayed on the CRT screen, the gray areas (alternate columns of pixels) will display in color. This is because of the way Apple makes hi-res color. Color changes can be made with any Apple paint program.



The Digital Paintbrush System

The Computer Colorworks

The Digital Paintbrush System consists of a high-resolution graphics input device and three disks of graphics utilities.

The designers at Computer Colorworks believed that the only natural way for people to draw is with pencil or pen on paper, so they created a new input device—the digital paintbrush. This device is an 8″ × 11 1/2″ inch plastic box that has a pen attached to a wheel inside the housing. The pen, with ballpoint tip, is attached by two strings that would appear to be in the way when you draw but in actuality are no problem at all. The unit plugs into the paddleport, and it can be used with other software in place of paddleport input devices such as joysticks.

You use the pen to draw on a paper that you've clipped to the box—as you draw, the software reads a switch that's activated in the tip of the pen. The drawing appears on the CRT as a standard hi-res image. You can replace the pen with a plastic stylus and trace images from books, cloth, and so on. The drawing area is restricted to a maximum of 10 1/2″ × 7 1/4″.

The software includes a Graphics Design Program, a Text Screen Editor, a Presentation Program and a Printer Dump Program. The Graphics Design Program has very useful options. One such option allows you to create a line, box, or circle and then move it around the screen until you decide just where you want to permanently "tack it." You can create curves by setting marks—the program connects the marks and constructs a curve as you watch. There are extensive color-mixing options. The menu, at the sides of the screen, disappears when your cursor touches it so that you always have the full screen to work on. A click of the button on the pen brings back the menu.

The software is excellent for the creation of technically oriented screens such as charts, floorplans, and so on. An area measurement program lets you draw or trace enclosed shapes and then recalculate the areas of these shapes. A utility disk contains fonts and library images to use in your graphics. Another disk contains an interactive telephone drawing program. With this program and a modem, you can draw interactively with another person who's miles away.

The KoalaPad with KoalaPainter

by Steven Dompier **Koala Technologies Corp.**

The KoalaPad package consists of a touch tablet and a high-resolution drawing and painting utility called KoalaPainter. The tablet, a 6″ × 8″ plastic device with a 4.5″ × 4.5″ pressure-sensitive surface, has two buttons. It plugs into the game controller port of the Apple computer. (See What's a touch tablet? p. 240.)

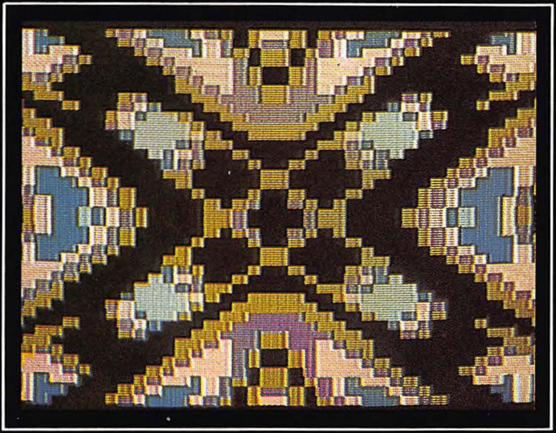
KoalaPainter presents you with a full screen icon menu. You move your cursor to the chosen icon and press the button on the tablet to make a selection. Your choices are Draw, Point, Line, Lines, Rays, Fill, Frame, Box, Circle, Disc, Erase, Storage, Mirror, Magnify, and Scale. You can draw with any one of eight brushes using any color from two color sets of nine colors. There are enough options here to create fine graphics.

The main problem with the KoalaPad is that random lines and dots erratically appear on the screen. This can happen even when you use the pad correctly, but for the most part, problems can be avoided if you observe these precautions:

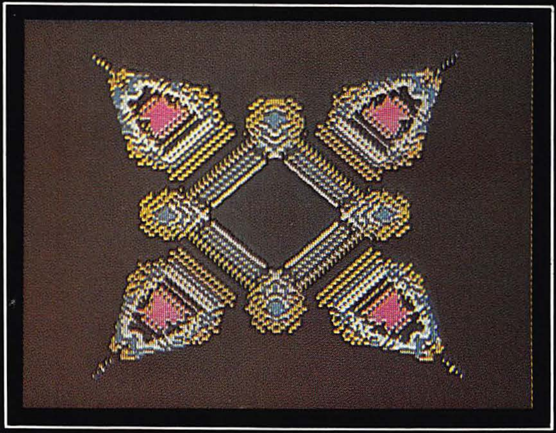
1. When drawing, apply a little more pressure than you find is necessary to draw with.
2. Keep the pressure consistent—don't lighten up as you draw.
3. Draw slowly—if you move the stylus too fast the tablet can't keep up with it and you might get erratic readings.
4. Some software, not specifically programmed to read input from the tablet, allows the cursor to "home" (jump to the upper left corner) as soon as stylus pressure is lessened. As the cursor homes it leaves a line in its path. You can compensate for this by pressing the button to turn drawing off *before* you lift the stylus from the drawing surface.

The KoalaPad can be used with most software that accepts input from joysticks or paddles.

Gideon Nettler



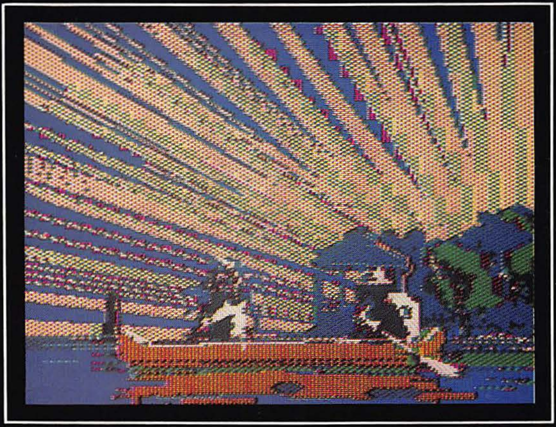
Symmetry Symphony 1



Symmetry Symphony 2

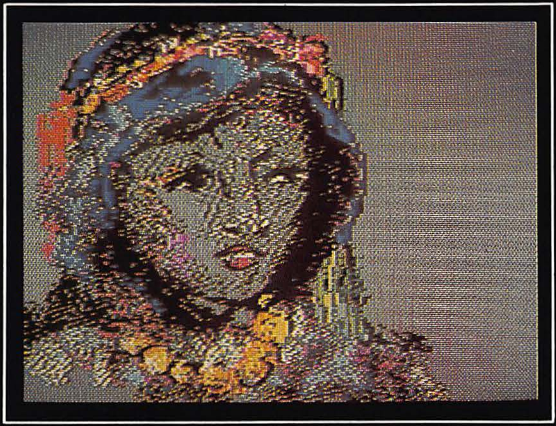
Rush Brown

Car in a wooded landscape

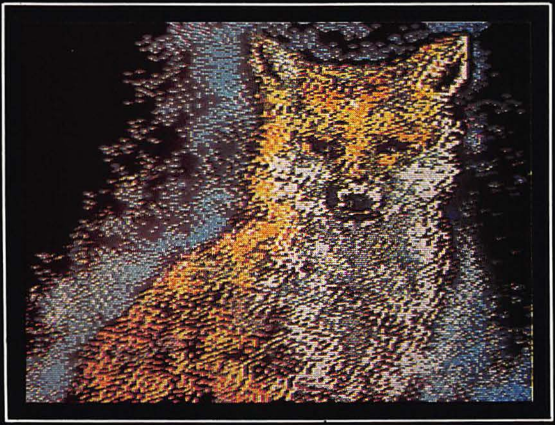


Canoe

Ame C. Flynn

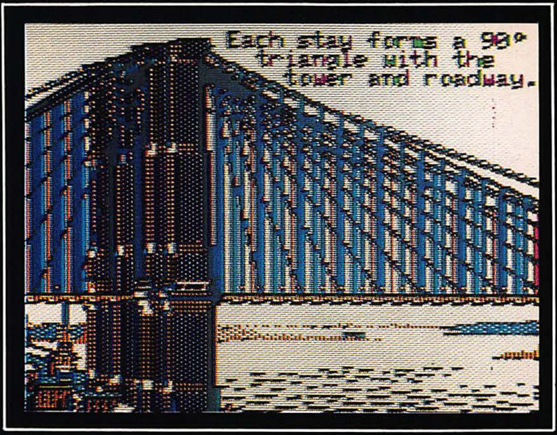


Berber

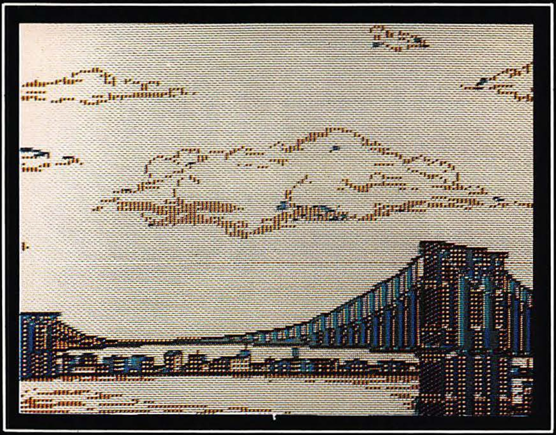


Fox

Jim Markowich

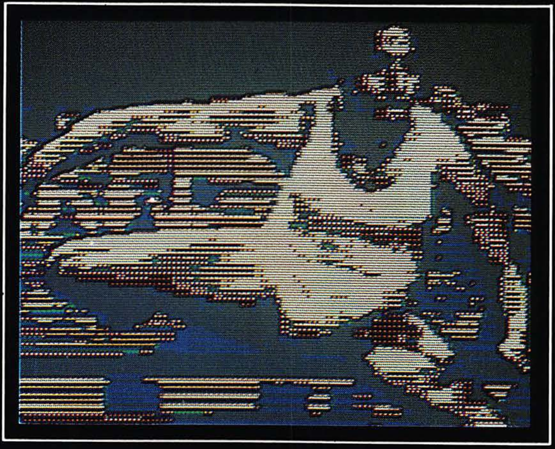


Each stay forms a 90° triangle with the tower and roadway.



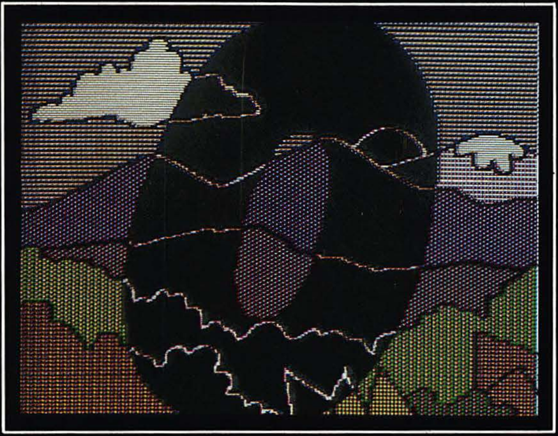
The Building of the Brooklyn Bridge

Howard Kessler



The Hurdler

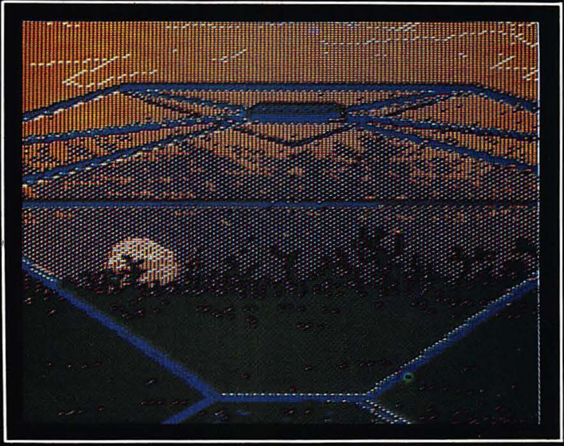
Lauretta Jones



October

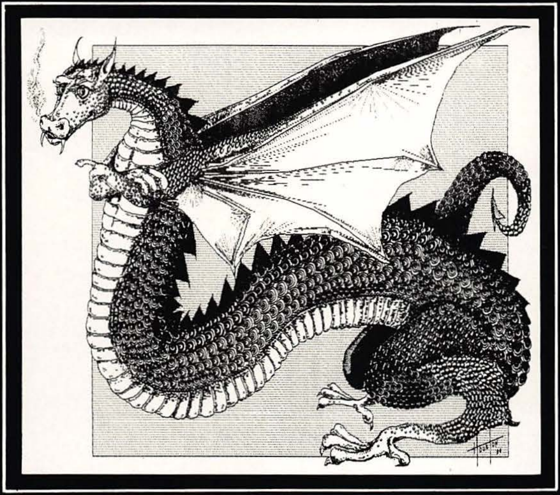
And we all fall down! (October)

Lorene Lavora



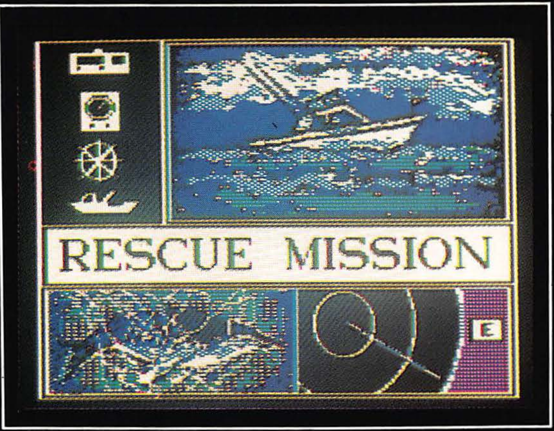
Domed Sunset

Duke Houston

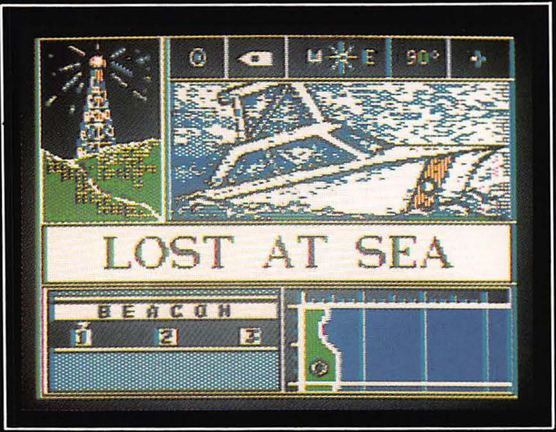


Dragon

Roberta Schwartz

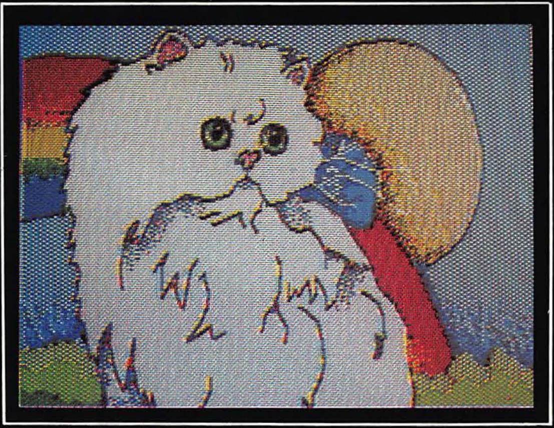


Rescue Mission - Title Screen

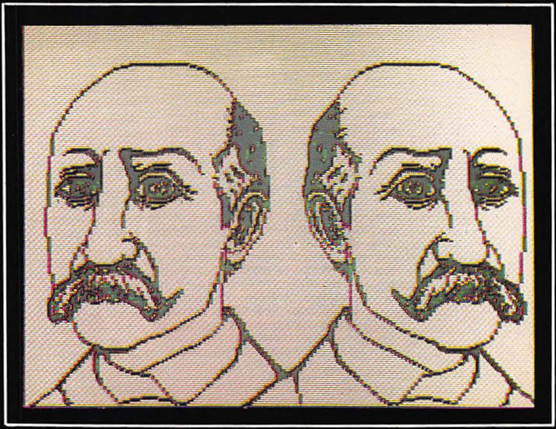


Lost at Sea - Title Screen

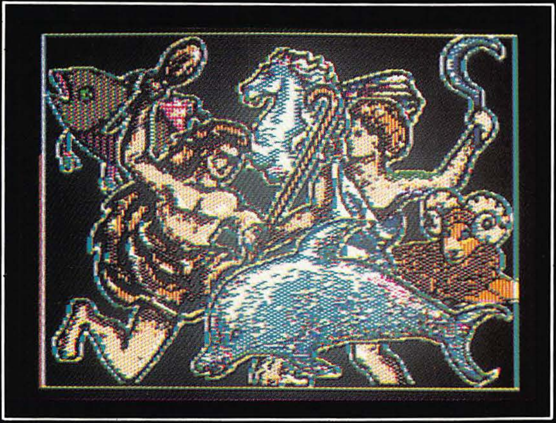
Reprinted courtesy of Bank Street College of Education



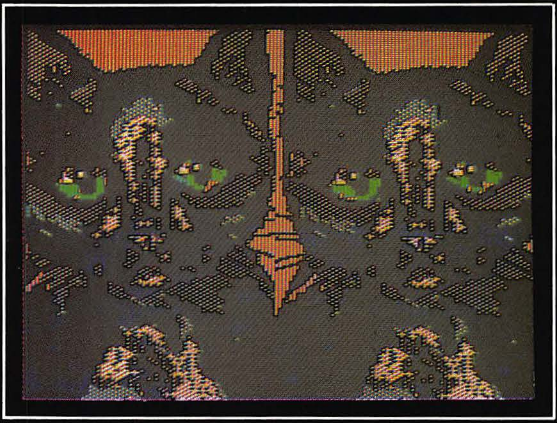
Over The Rainbow



About Face



Night Sky ***(Reprinted Courtesy of Helicon Productions)***

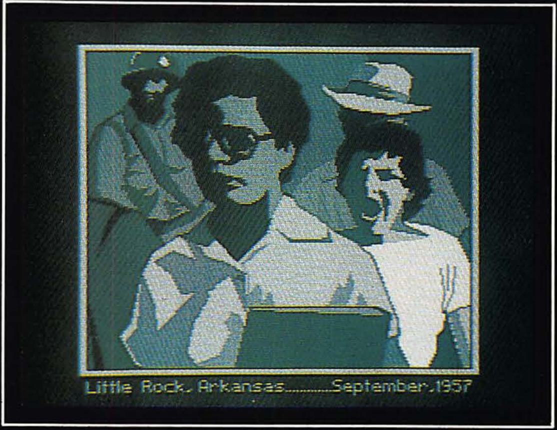


Twins

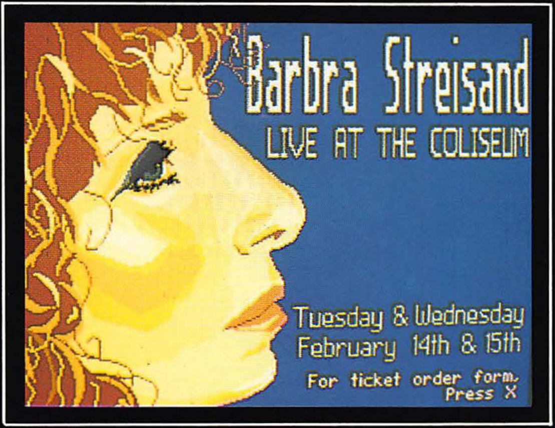
Maria Manhattan

It's
APPLE
BYTES
!!!

Apple Bytes

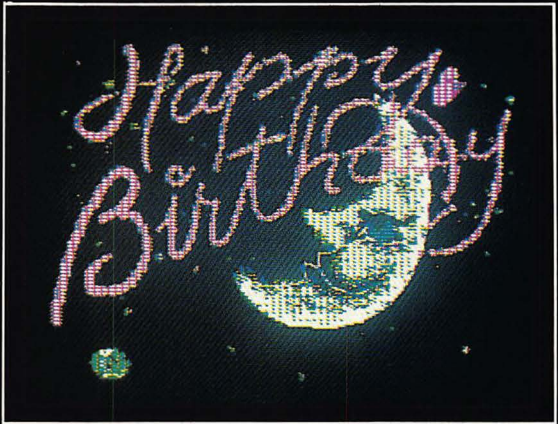


Little Rock, Arkansas . . September, 1957



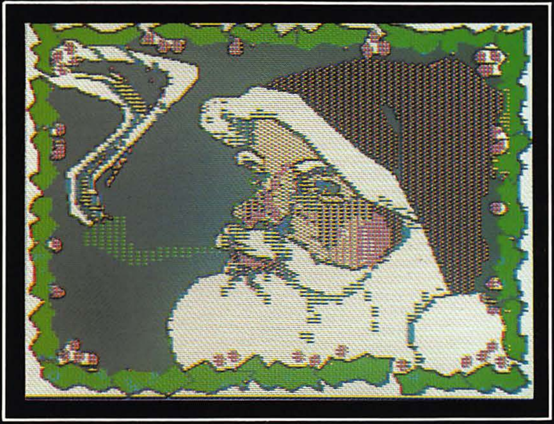
Barbra

M. Brooks Jones

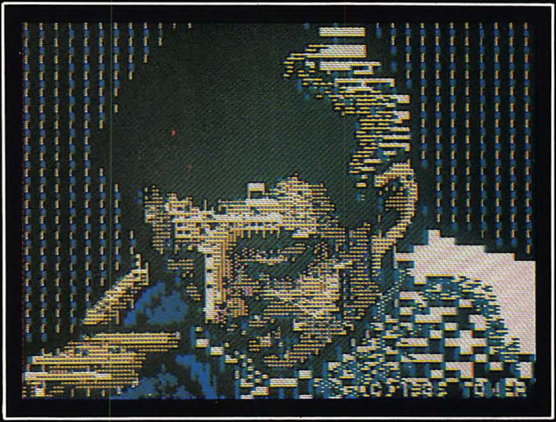


Lunar Birthday

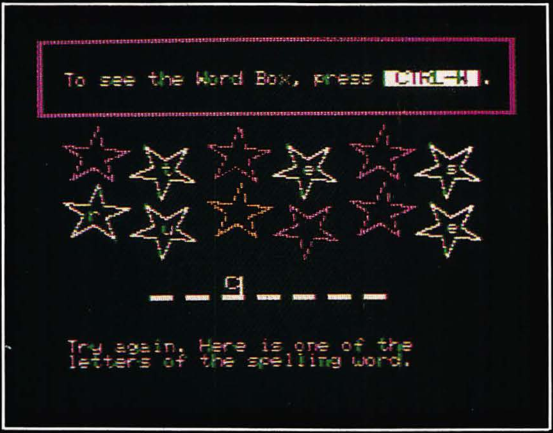
Catherine Tower



Santa Claus



Michael Callery

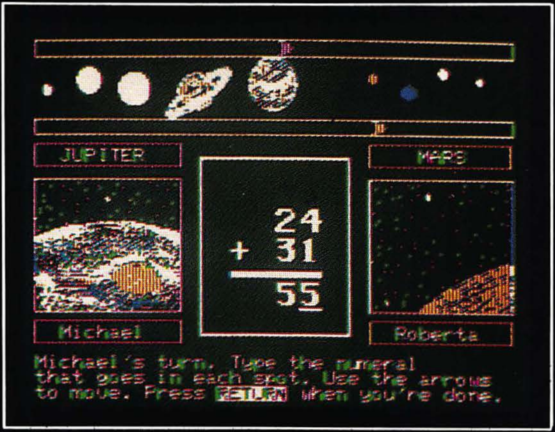


To see the Word Box, press CTRL-W.

_ _ g _ _ _ _

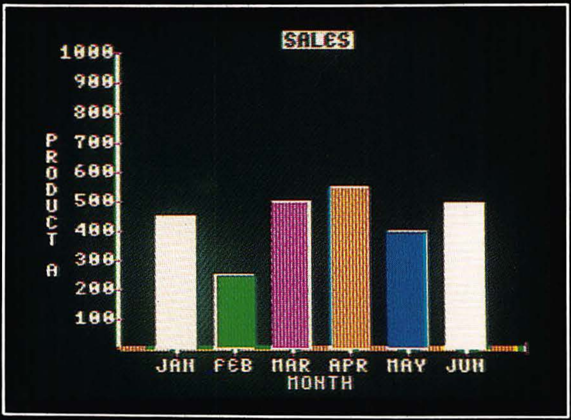
Try again. Here is one of the letters of the spelling word.

HBJ Starshapes - HBJ Spelling Practice
Orange level 4 Part 2



Michael's turn. Type the numeral that goes in each spot. Use the arrows to move. Press RETURN when you're done.

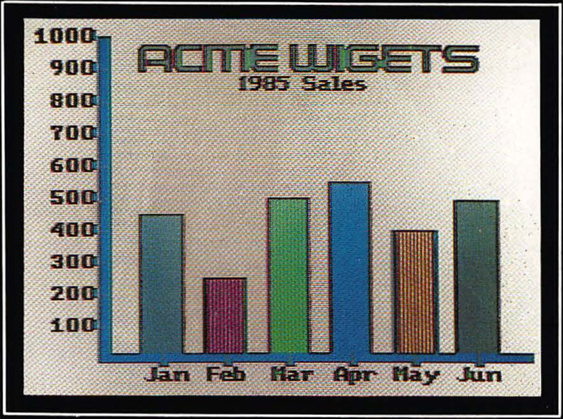
HBJ Planet Hop - HBJ Mathematics Today
Practice Diskettes levels 5 - 6
Purple/Brown Part 2



Bar Chart 1

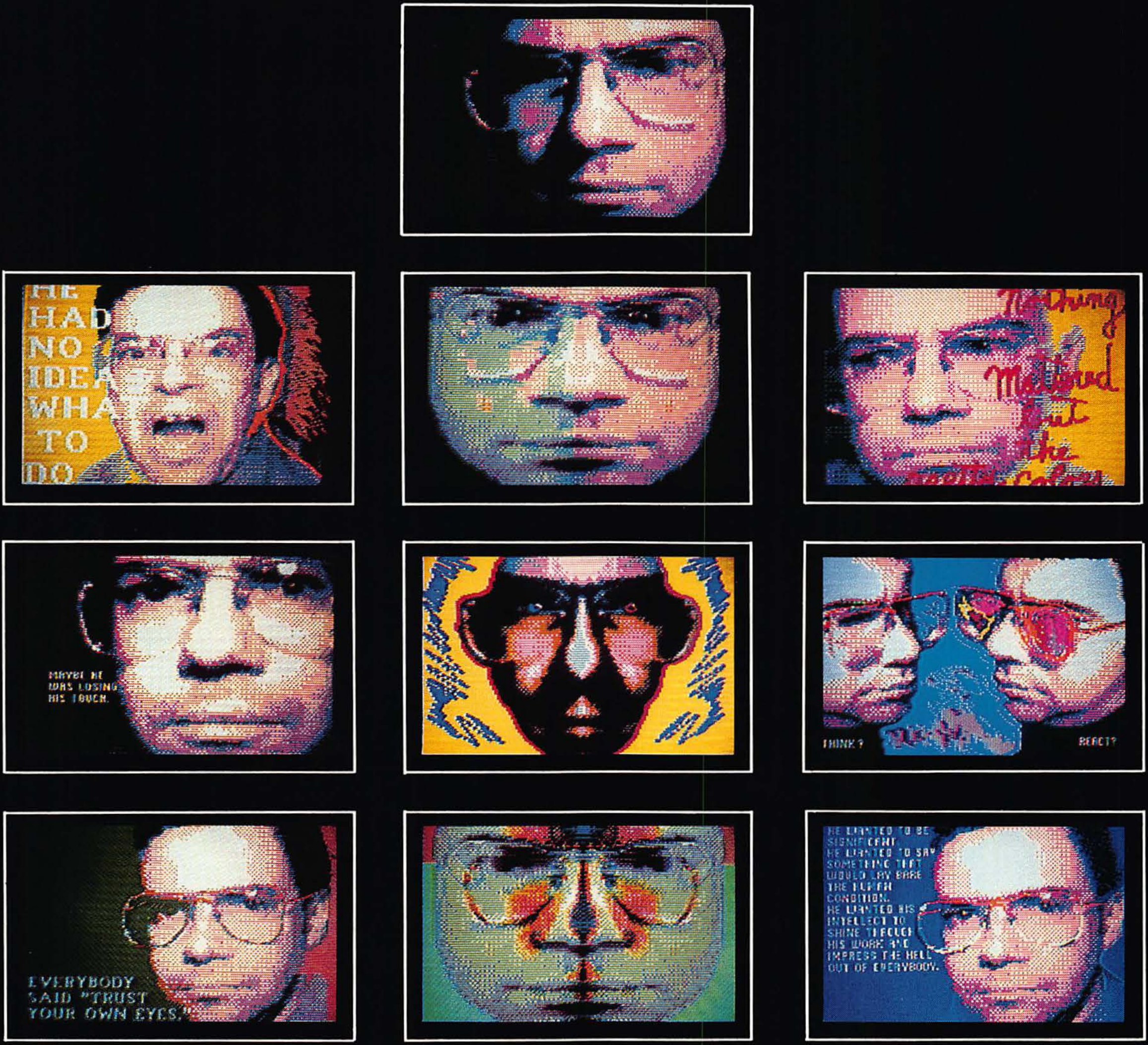
A presentation graphic created with The Graphics Department. The screen was then brought into a paint program for enhancement.

The final graphic. First the labels and the font were changed, and then the entire graphic was reversed.



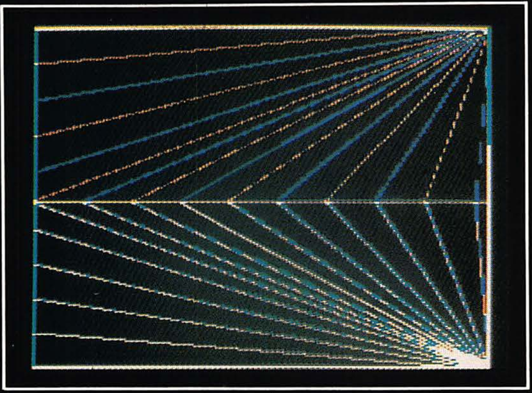
Bar Chart 2

Ken Glickfeld

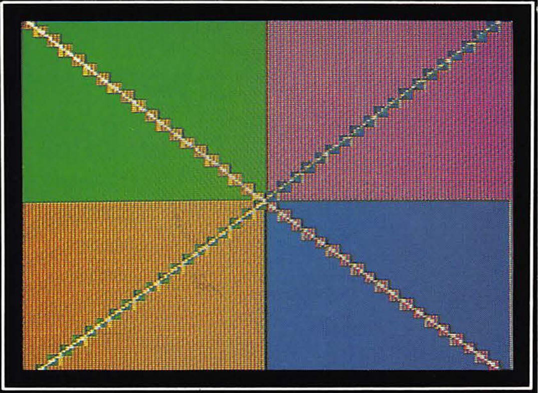

HE HAD NO IDEA WHAT TO DO
Nothing Mattered but the pretty colors
MAYBE HE WAS LOSING HIS TOUCH.
THINK?
REACT?
EVERYBODY SAID "TRUST YOUR OWN EYES."
HE WANTED TO BE SIGNIFICANT HE WANTED TO SAY SOMETHING THAT WOULD LAY BARE THE HUMAN CONDITION. HE WANTED HIS INTELLECT TO SHINE THROUGH HIS WORK AND IMPRESS THE HELL OUT OF EVERYBODY.

Left Face

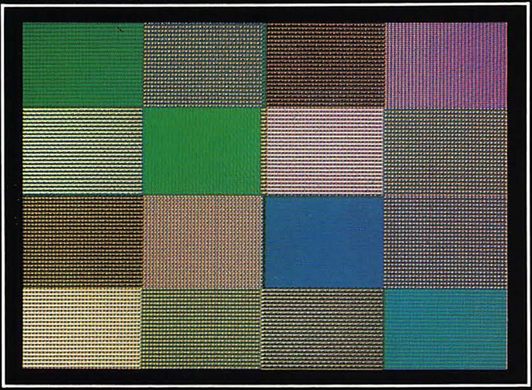
Apple high - resolution graphics



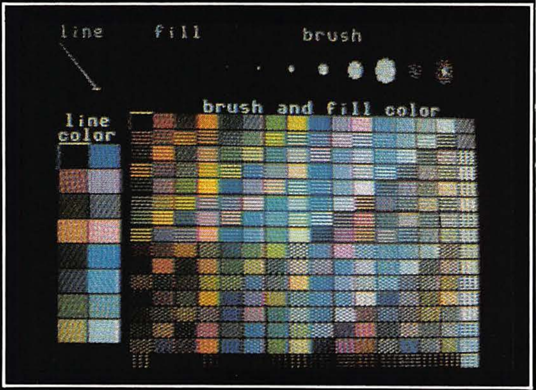
Aliasing. Only horizontal, vertical and 45° diagonal lines are smooth. Other lines are jagged.



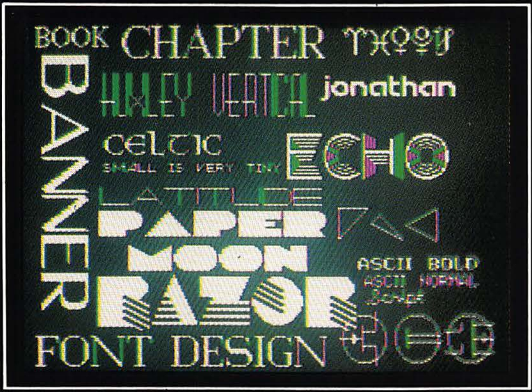
Color Problems. Lines of white from one color set crossing colors of another set.



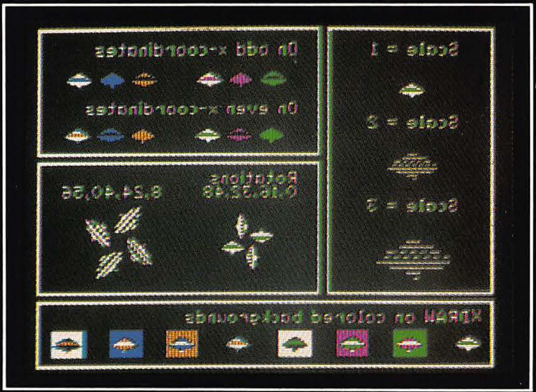
Dithering. Apple's hi-res palette expands when colors are interleaved.



The menu and extended double-high-resolution palette from The Complete Graphics System.



Fonts. A sampling of the wide selection of available fonts. These are from Fontrix.



Applesoft shapes. Shapes can be scaled, rotated, drawn and X-drawn.

The Apple Graphics Tablet with Tablet Software

Apple Computer Inc.

The Apple Graphics Tablet package consists of a magnetic tablet with interface and stylus (commonly referred to as a pen), mylar plastic overlay, and software. The tablet reads input from the stylus. (See What's a graphics tablet? p. 242.) Drawing with the stylus is as natural as drawing with pencil on paper.

The software that comes with the Apple Graphics Tablet is a high-resolution drawing and painting program called, simply, Graphics Tablet Software. To use the software you must first attach the tablet menu, a mylar overlay, to the tablet. The overlay menu is divided, gridlike, into program options. You make a selection by pressing the stylus on the option of your choice. Lines, boxes, circles, frames, dots, color changes—all are accessed with a touch. The program doesn't require you to put down the pen and enter commands via the keyboard, which makes it especially comfortable to use.

Some special features of the software include Window, which lets you draw something very small and have it appear enlarged on the screen (good for tracing); Viewport, which confines your drawing to a limited area of the page thus protecting the rest of your image from any "accidents"; Slide, which lets you slide your image around the screen; Reducer, which lets you convert large drawings to smaller images; Separate, which lets you create color separations of your image; and Delta, which gives you control of the pen's drawing precision.

The Apple Graphics Tablet is probably the best graphics input device available for the Apple computer. When it was first introduced, its accompanying software's features weren't available anywhere else. Today, this software is still excellent but no longer unique. In fact, we prefer using another package called The Designer's Toolkit, with our tablets. You can find out more about that program in the software section.

The Apple Mouse II with MousePaint

by Bill Budge **Apple Computer, Inc.**

Currently, there are two Apple Mouse II packages that are functionally identical to each other. The particular Apple II computer you are using determines which package you would need. (See What's a mouse? p. 243.)

The Apple Mouse II works flawlessly once you've grown accustomed to drawing with this device. We've used the Mouse II with MousePaint as well as with other compatible graphics software; in all cases the mouse was fast and accurate.

The software that comes with the Mouse II is a high-resolution paint utility called MousePaint. Designed to emulate the Macintosh's MacPaint software, it has a similar screen display with pull-down menus and many of the same graphics options. The program doesn't compare to MacPaint, however, and is in fact not as good as many of the paint utilites currently available for the Apple II line of computers. You can use only black when you work with the brushes, and there's no fill option. You don't see the entire image when you work on a graphic because the menus on the left and bottom take up part of the screen. Also, as of this writing, MousePaint supports only the Apple line of printers for hardcopy output.

MousePaint operates under ProDOS, so if you want to combine graphics created with this software with graphics created with software that operates under DOS 3.3, you'll have to convert the images.

Fortunately there are many excellent software programs that can use the Mouse II for input, including all of the latest double-high-resolution software.

What kinds of graphics output devices are there?

The television (or CRT) screen is the Apple's primary output device. When you draw and paint, your picture appears on this screen; it can, therefore, be thought of as an electronic sketchpad or canvas. However, just as the traditional artist can elect to create art in media other than paper or canvas, such as fabric, clay, glass and so on, the screen isn't the only output for computer art. An abundance of peripherals are available to let you create just about any type of output you might need. In general, the higher the quality (in either resolution, color, or both) of the output, the more expensive the peripheral and the software needed to drive it.

For paper output, inexpensive dot-matrix printers can yield good black and white printouts or fair to good color printouts for less than $500. For video output, multicolor ultra-high-resolution devices may cost $5000 and up. As with any peripheral you must be certain that the software you're using supports the external device.

Peripherals for specialized output fall into several general categories:

Printers	paper hardcopy
dot-matrix	graphics: b/w or color; general purpose
letter-quality	typewriter quality text
ink jet	graphics: b/w or color; general purpose
laser	graphics: b/w; typeset quality text
Plotters	business graphics; CAD/CAM; 3-D
Cameras	slides, film, videotape
Ultra-hires	video

Beginning computer graphics artists should not invest in a multitude of output devices until they determine what their needs are. The only peripheral on this list that we would consider a "must" is a printer with graphics capability. (See What's a printer? p. 21.)

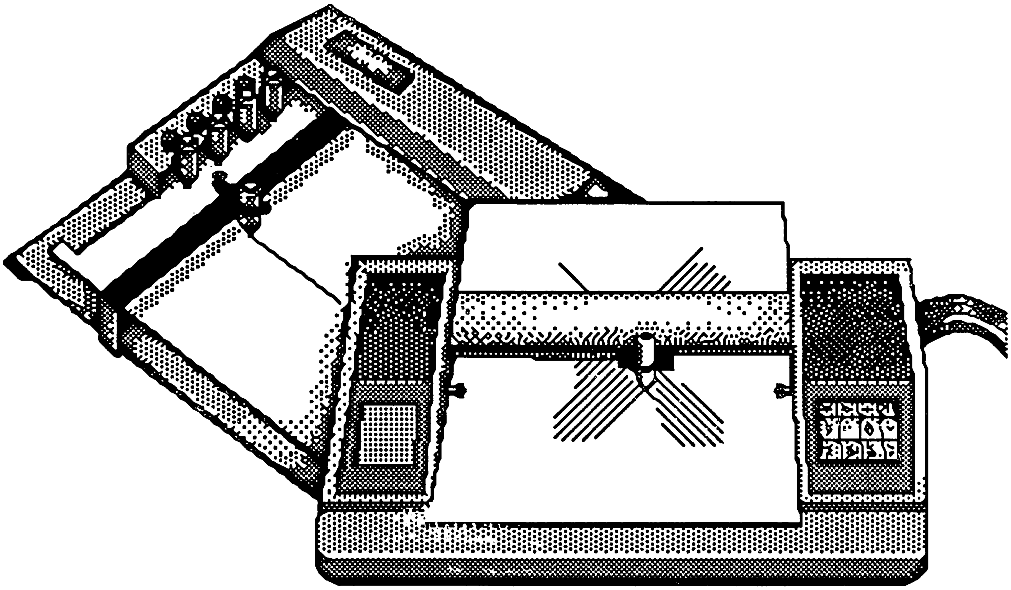
What's a plotter?

Like printers, plotters produce paper hardcopy. Printers, however, are character-oriented devices while plotters are graphics-oriented devices. Plotters cannot easily reproduce the Apple's graphic screen because the screen is a grid of dots—plotters draw lines.

A plotter may have one, two, four, or more changeable pens (usually felt-tip color pens), which are moved over paper by a mechanical arm. The arm is directed by the computer via plotter commands such as pen up, pen down, move to coordinates 1000,1000, draw the letter "A" at a scale of 2, and so on. Specialized plotter software may or may not produce an image on the graphic screen as it draws it on the paper.

The most common applications for plotters are business and presentation graphics and computer-aided-design. These fields require the highly precise, detailed graphics that a plotter can provide.

If you need highly detailed linear graphic output you may want to invest in a plotter. Be sure to check the software you'll be using to see which plotters it supports. Most plotter manufacturers have their own set of plotter commands; software that drives a Hewlett-Packard plotter may not drive a Houston Instruments plotter.



How does a printer print graphics?

Printers print graphics by transforming the printed page into a large block of dots. By stretching the terms a bit, you can speak of printer pixels and a printer raster block. The printers best equipped to do this are those already using a dot matrix technology. (See What's a printer? p. 21.)

In the normal operation of a printer, information coming in the serial or parallel port is interpreted as being ASCII characters. The printer merely puts these on the paper. Some of these characters may be interpreted as printer control codes to accomplish special effects like boldface or italic fonts.

Many printers also have a code to turn on graphics mode. In this mode, the information coming into the printer port is interpreted as pixel information. A printer with eight wires in the print head might use the bits in a byte to indicate which wire to actually print with. Most printers have more than eight wires in the print head and have more complex graphic commands.

The memory holding the Apple graphic screen contains information that's interpreted by the Apple video circuitry as screen graphic information. No printer we know of can use the screen data directly to print an image. Instead, a program must translate the Apple's screen into printer graphic codes. Because every manufacturer has used different codes, nearly every printer requires a different translation program. This program can be contained in ROM on the printer interface card or in a separate graphics dump program.

If the printer graphics translation program is in ROM, you can print graphics from your own programs or directly from Basic. (See How do I print graphics? p. 170.) If the program is a RAM-based program, you may have to save your graphic as a standard Apple screen and run a printer dump program.

It's possible to achieve much higher resolution on the printer than on the Apple screen. Graphics printers may feature resolutions of 160 × 140 dots per inch or higher. Because a program translates the screen or other graphic information, this higher resolution is accessable with the correct software. Fontrix, by Data Transforms is such a program.

Can I make slides from Apple graphics?

There are several ways to make slides from Apple graphics.

The simplest and least expensive is to photograph the screen yourself. Presently, this is your only alternative if you want slides of low-resolution or double-resolution graphics. (See How do I photograph the screen? p. 253.) Unless you're an accomplished photographer, the quality of the slides may not meet your expectations.

One way to get high-quality slides of standard high-resolution graphics is to purchase a photographic peripheral. These peripherals range in price from expensive to very expensive.

One type of photographic peripheral attaches to the computer via a serial port and contains a high-resolution black-and-white video screen. This device uses the information coming from the computer to reconstruct your screen image on its own video screen. A fixed-focal-length camera then photographs that video screen through a series of filters. Multiple exposures result in a full-colored image. By manipulating the filters and the information being sent to the device, it's possible to generate colors not normally available on the Apple hi-res screens.

Other photographic peripherals, like plotters, produce their own graphic rather than reproducing the Apple's screen. These devices offer very high resolution (1024 X 1024) and a full selection of fonts for labeling. Circles and diagonal lines don't alias and there are few limitations on the numbers and hues of colors. These systems are very costly and are specifically geared for business or CAD/CAM graphics—they may not support fine or commercial art applications.

If you don't have confidence in your photographic skills, there is another alternative to investing in an expensive photographic peripheral. You can send a disk of standard hi-res graphics to a company that has such a peripheral and specializes in creating slides from computer images. One such company is Computer Slide Express, also known as Visual Horizons, in Rochester, New York. At the time of this writing, the fee for this service is $6 per slide, with a minimum order of $30.

How can I photograph the screen?

Techniques for photographing the CRT display are as varied as the photographers. Generally, it's a trial-and-error process.

Many computer artists, unsatisfied with the quality of slides and prints photographed directly from the screen, invest in expensive peripherals or send their disks of graphics to companies that have specialized equipment for this purpose. (See Can I make slides from Apple graphics?, p. 252.) At this writing, the advent of double-high-resolution has changed the picture because the specialized equipment used for conversion of disk-based images to slides can't accomodate double-hi-res graphics. We're clicking our shutters again.

Here are some guidelines for photographing the CRT display:

Required:

Quality, nonautomatic, 35mm single-lens reflex camera
Tripod
Cable release
Film—Kodachrome 64 or Ektachrome 64 or 200.

Optional:

Close-up or telephoto lens—it can make the difference!
Commercially available hood that fits on the CRT screen

1. Take your pictures in a darkroom or use a hood over the screen to prevent light from reflecting on the screen.
2. Set your camera on a tripod and adjust it to the correct height of the CRT display. Align the camera squarely with the screen. You may get better results if you close-in on the image to avoid the distortion that often occurs in the four corners of the screen. This is where a close-up lens helps. If your images come out distorted, try a telephoto lens in the 150-200mm range.
3. Set the display to medium brightness.
4. Bracket all shots—usually between f/4 to f/8. Try different shutter speeds between 1/30 second to 1 second. It takes the electron beam in the CRT at least 1/30 second to scan the entire screen.
5. Keep a written record of the f-stops and shutter speeds, so that when you process the film you'll know which exposures were the most successful.

Many factors influence the success of your efforts. The quality of the CRT display is a major one.

Can I make movies of my graphics?

Easily! Just photograph the screen with a super-8 camera. Use a tripod and shutter release to keep the image steady. We've gotten good results with Kodak Ektachrome film but you should experiment with films and filters to get the color balance you want.

Can I videotape my graphics?

Yes, but not without problems, because Apple's video signal is compatable with, but not identical to, the American television standard (NTSC).

One method is to attach the video output of the computer to the video input of a VCR and record the signal, but you may get poor quality and/or only black and white video. Higher-quality cables and sometimes shorter cables may result in better images.

Another method is to use an RF generator on your Apple, connecting it to the antenna input on the VCR. The image will be less clear than with a direct video connection, but it will be in color and should be of decent quality. Don't count on editing videotape taken directly from an Apple—the nonstandard signal will play havoc with video editing equipment.

To produce videotape for editing, there's a peripheral which converts the nonstandard Apple signal into a standard NTSC video signal. Inexpensive converters fit into one of the expansion slots in the Apple II, IIplus, or //e and restructure the Apple's video to NTSC specifications. The more expensive converters have several circuit boards or are stand-alone systems connected to the Apple by a cable. These devices not only create a standard NTSC signal, they also allow you to manipulate colors, mix a live or videotape signal with the Apple's signal, and achieve other special effects.

What's ultra-high-resolution?

Apple's built-in graphics include lo-res and hi-res, as well as double-lo-res and double-hi-res on Apple //c and 128K Apple //e's. Some applications require higher resolution or more colors than are available in these modes—an external graphics processing unit is the answer.

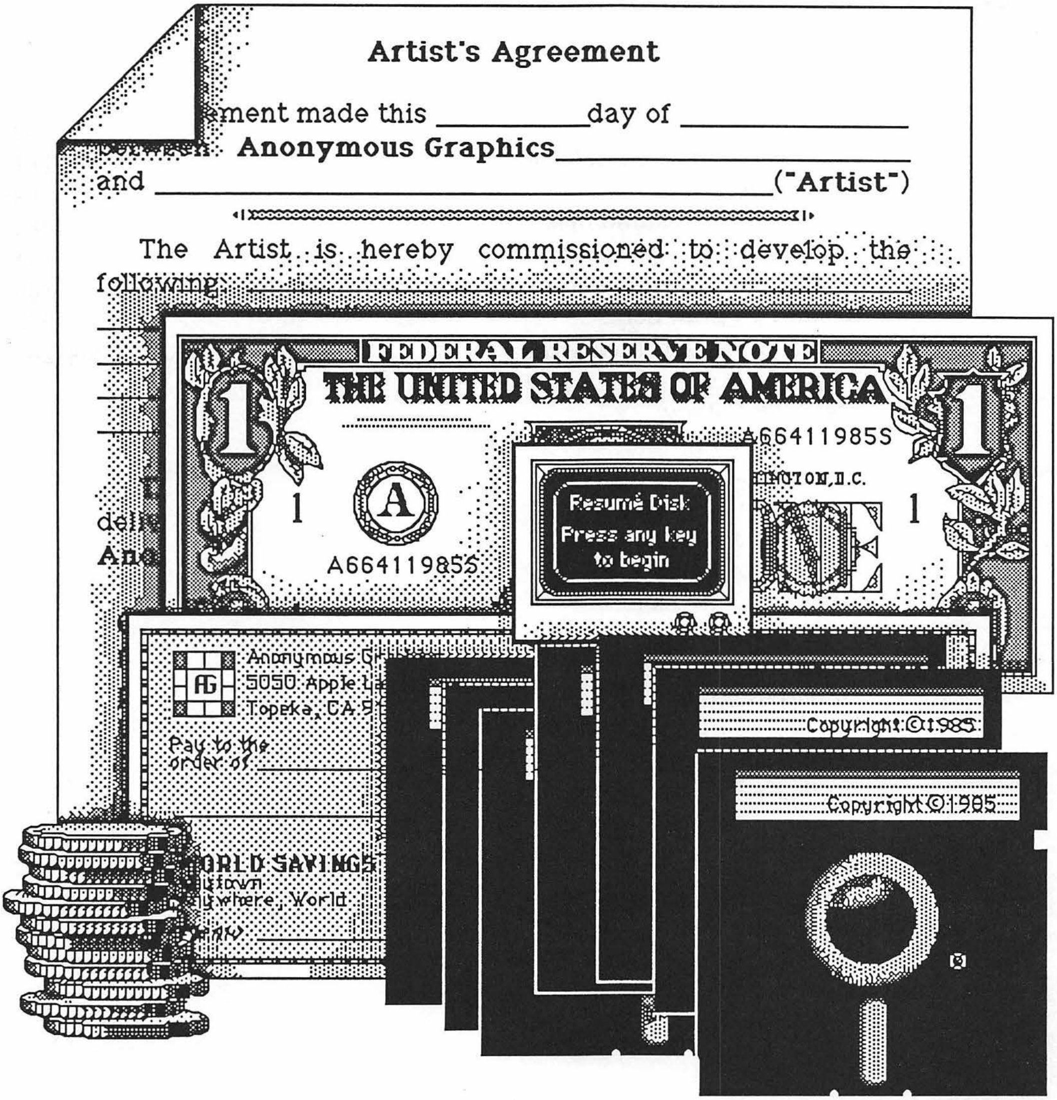
With an ultra-hi-res graphics processor, such as the Number Nine System, you can get 512 by 512, or 1024 by 1024 resolution and a display of 16 colors from a palette of 4096, or 64 colors from a palette of 16 million!

This unit is viewed by the computer as an external terminal. For the most part, these ultra-hi-res systems are full computer systems lacking only in I/O. A program, running on the Apple, sends the external unit a command—draw a line from 0,0 to 512,400; fill at 200,400; or draw a circle at 350,350 with a radius of 50—and the unit executes the command. Unlike the built-in graphics of the Apple, external graphics units require no Apple memory to store an image. Therefore, programmers have more RAM for sophisticated programming.

Ultra-high-resolution systems are ultra-high-priced systems (approximately two to three thousand dollars), and after the purchase, using them incurs additional expenses. One popular system can store only one image on a disk, and the images can only be displayed on expensive, high quality RGB monitors.

These systems require specialized programming, so it's usually difficult to find good software for them. Some of them are equipped to convert standard Apple high-resolution screens to ultra-high-resolution screens. Like standard Apple graphics, you can produce paper hardcopy, slides, and video from ultra-hi-res graphics.

Depending upon your needs, a quality ultra-hi-res unit for the Apple can be a worthwhile investment. It provides graphic tools that would cost many thousands of dollars more as stand-alone units.



About careers and the artists

About careers and the artists

What about jobs?

Assuming you like computers, assuming you like art, and assuming you can create professional-quality art on a computer, there are two basic choices as you consider a career—micros or mainframes.

If you find yourself loving the one-to-one relationship of microcomputers, find yourself cancelling engagements because you'd rather be shifting shapes, find yourself staying up all night playing with this most incredible machine, you may want to forge a career as a microcomputer artist.

Although microcomputer software and hardware firms, Apple included, do have graphic artists on hand to design and execute graphics for their products, microcomputer graphics are as yet an embryonic field. Few traditional design studios have made an effort to incorporate micrographics in their repertoire of services. This is changing. With the Macintosh and the latest generation of graphic-intensive microcomputers, artists should be in increasing demand at computer companies.

For the moment though, most microcomputer graphic jobs are freelance. Because micros are being used in nearly every aspect of business and education, these industries need graphics to accompany their software products. The jobs range from small—translation of a logo to a computer graphic—to massive—developing a half-hour marketing videotape. You may, however, find it difficult to earn a living with your graphics.

At first, you may have to maintain another job or freelance business. But once you've developed expertise you may find yourself creating in-house business graphics for a Fortune 500 firm, designing software that will teach five-year-olds how to identify shapes, teaching computer graphics in a local school or college, writing articles on graphics for magazines, or even developing your own software line. We know; we've been there.

The best advice we can give people who want to stay with microcomputer graphics is to learn. In an industry in constant change, you must keep on top. Read magazines and books, visit computer stores, try new hardware and software, and take courses. Most important, work! Master drawing with every graphics input device. Create an animation; create ten. Draw things and use utility programs to generate as many variations of the images as possible. Seek out other artists and graphic programmers. Join the network. Learn, practice, and connect with your colleagues.

If aliasing really bothers you, if you feel you must have a palette of 15 zillion colors, if you envision your name on the credits of Star Wars 7, you should broaden your job search to include mainframe, broadcast, and print computer graphics.

Your experience with the Apple has taught you most of the essentials, not only terminology but also the concepts. Every computer graphic system is different and you'll need training in the particular graphic system a company uses, but the training can be "how do I . . .?" instead of "what, why, and how do I . . .?" If you were hiring an artist, which one would you rather hire?

Your Apple resumé disk is as important in the larger computer graphics field as it is in the microcomputer arena, for it's the only way an employer can decide if your skills are suitable. However, if you're seeking a job on a larger system, you should try to get time on that or a similar system. Some universities have computer graphics labs that you may be able to use. If you live in or near a large city, there may be a local SIGGRAPH (Special interest group) chapter where you can meet artists and make connections. You may also benefit from attending the annual conventions of the National Computer Graphics Association or SIGGRAPH to see equipment and meet colleagues and prospective employers.

Whether micro or mainframe, we don't recommend sending a resumé disk to a company without some advanced leg, paper, or telephone work. First, do your homework: what kind of work does that company need? What products have they published? How would your work improve their products? Find out if the company reviews resumé disks. Identify the individual within the corporation who's in charge of interviewing artists and request an appointment. If you do arrange a personal presentation, be sure a microcomputer will be available when you visit or you'll have to bring your own (this is known as the //c advantage).

Hobby or career, micro or mainframe . . . we wish you well.

What kinds of jobs are available?

Microcomputer graphics is largely a freelance career now, but this will change. With talent and creativity, you can forge a rewarding career. The following list is not exhaustive, but it illustrates some of the the opportunities for work available to the Apple artist.

General graphics and design
Logo design or translation from other media
Custom font design
Icon design
Screen design

Games/Software
Illustration of adventure games
Creation of game play screens
Animations
Title screens

Publishing
Cover design
Line illustration
Layout

Business
Annual report covers
Interior illustration
Presentations—print and CRT based
Newsletters
Advertising
Business graphs and charts

Videotex

Architecture/Mechanical Design (CAD/CAM)

Fabric/Textile/Crafts
Textile design
Needlepoint design

Fine Art
Portraiture
Conceptual pieces
Interactive pieces

What's Videotex?

Videotex is a technology which combines computers and telephones to produce a new information utility. As the name implies, text is presented on video.

Typical use of a videotex system might be to make theatre reservations. After turning on your videotex computer (which would combine television, telephone, and the computer) and dialing the mainframe computer housing the videotex information, you'd be greeted with a log-on sequence. This sequence would assure that no one else had access to your information or charge accounts. Once accepted by the system, you'd be presented with a menu listing the primary videotex services, perhaps MAIL, SHOPPING, EVENTS, BANKING, and NEWS. If you were an old hand at the service, you could bypass other menus and go directly to the theatre reservation services. New users might select EVENTS and then be presented with another menu: MUSIC, SPORTS, THEATRE, DANCE, SPECIAL, and SEASONAL. Selecting THEATRE would present you with a list of shows. Selecting the show you wanted tickets for would generate a series of questions about seats and dates, and then you'd be able to purchase your tickets.

One of the things that makes videotex so exciting is that all along the way you would be interacting with graphics. In the scenario above, for example, you'd be presented with a seating plan with all the available seats highlighted. All the menus would be illustrated, some even animated. Eventually, as videotex is merged with videodisk technology, you may even be able to see a snippet of the show.

Videotex has not yet become popular—there are many technical and social issues unanswered. Some of America's largest corporations, however, are staking millions of dollars on its eventual success. Videotex standards have appeared from other countries—Canada's NAPLPS system seems to have the edge in the U.S. over the British PRESTEL system. There are NAPLPS interface cards that enable the Apple to create and receive frames created with this standard. Although the videotex system of the 1990s may more resemble a telephone than a personal computer, today's Apple can and is being used for frame creation on videotex systems.

The knowledge you gain working with graphics on an Apple computer can be the key for employment on larger, more complex systems.

How do I make a resumé disk?

Your resumé disk is part of a portfolio which may consist of printed hardcopy, samples of work already published, and so on. Like the other components of this portfolio, the resumé disk should show work that's been published, if any, as well as samples of the type of graphics that you do best.

How are the graphics at which you're skilled being used? Illustrate an imaginary, or existing, adventure game; create a font to display mathematical formulae; design an iconic menu; design book covers; or create cartoons. Your work must be as polished and professional as it would be for an actual client. Be careful of color problems and clean up errant pixels.

Once the graphics are completed, you'll need title, credit, and copyright screens. If you're using animation, packing or other routines from commercial software, give credit to the program. Check the documentation; some publishers have a required format for credits. (See Can I use others' routines in my work?, p. 266.) If you don't own the copyright to your graphic, be sure you obtain permission from the copyright holder before using it. If you do own the copyright, include your copyright notice on the disk.

You'll need a program to merge your graphics into an auto-running disk presentation—don't expect a prospective client to do more than insert a disk in the drive and turn the computer on. The graphics must be displayed elegantly.

For hi-res images, use a slideshow utility that packs the pictures and provides a choice of wipes and dissolves to give your resumé disk real polish. If you have a mixture of animations, lo-res, and hi-res screens, you'll need a custom program to display the graphics. You may want to get a professional programmer to create an effective display program. A local Apple User Group is a good place to find such an individual—you may be able to barter graphics in return for programming help.

One other point: you may not be able to create stand-alone graphics with certain copy-protected software. It's disconcerting to explain to prospective clients that they can't see your work unless they first boot a particular graphics editor. We avoid such programs.

How can I protect my work?

A contract is a binding agreement between two or more parties. It can be in the form of an oral agreement, a purchase order, a confirmation form, a letter, or a formal document. Although oral contracts are binding in many cases, we recommend you always have written contracts.

If you're employed as a computer graphic artist, chances are the company you work for will own the rights to all work you create for the firm. Your employment contract should specify your rights.

If you're a freelance artist, you have options. Initially, you own the copyright to all your work. Upon receiving a commission, you'll be offered one of two standard types of contracts. In the first, a work-for-hire contract, you relinquish all rights to the work. In a non-work-for-hire contract, you're essentially loaning the work to the contractor for a specified use over a specified time period.

Most contractors prefer work-for-hire contracts. Many computer graphic artists, like artists in other fields, are refusing work-for-hire contracts. Organizations like the Graphic Artists Guild are trying to make work-for-hire contracts illegal.

The very nature of electronically generated art brings up new contractual problems. Before you sign a contract, be sure that the following points are addressed.

1. What can the client expect in the work? Because of color and resolution limitations you may not be able to exactly reproduce a logo or other image that wasn't originally designed with Apple graphics in mind. Be sure your employer is aware of the characteristics and limitations of the media.
2. What constitutes completed work? Although you're the artist, some clients won't defer to your expertise and have strong ideas about how things should look. They may demand numerous revisions. Establish how many revisions and what kind of revisions you will do without additional payment.

3. Who owns the rights of translation to other computers? Some forms of Apple graphics—those created with the Graphic Magician picture painter, for example—are readily transferred to other microcomputers. After the transfer, the pictures may need touching up, color substitutions, and other revisions. You should receive a royalty or payment for the translated graphic. Also, you should be given the option to do all revisions of the graphic before another artist is called in. Be sure to clarify your client's plans in this area.
4. Who owns the rights of translation to other media? Computer graphics are easily photographed, videotaped, and so on. Do you want your graphic to appear on billboards across the country without additional payment?
5. Will you receive credit for your work? Credit for computer graphics can be given in a variety of ways—on the disk that displays the graphic, on the documentation that comes with the disk, on the packaging, on the hardcopy reproduction of the graphic, and so on. One of the problems many artists are having involves publishing houses that commission graphics for software and then use sample screens for advertising without giving credit to the artist.
6. Establish in your contract that any electronic alteration of your work, (exchanging colors, flopping, merging, magnifying, deleting, and so on) creates additional art, therefore constituting additional use, and should be negotiated for separately. There's no precedent for a medium that can be altered with one keystroke. Protect yourself.

Can I use others' routines in my work?

Apple full-screen images are 8K or 16K bit maps. After you use a drawing or paint program to create these images you use only the image you created. Generally, there's no need to give credit to the programs you've used to create a still, full-screen graphic, although you may want to do so out of courtesy to the author or publisher.

Graphics routines, such as those used for animation and other special effects, are a different story. To use these, you have to copy the routines to your own disk and use the code to display your graphics.

If these routines are in the public domain, you're free to use them, but you must follow any credit requirements of the author of the program. Program documentation will usually specify these requirements.

If the routines were obtained from magazines or other computer publications they are not in the public domain—they fall under the same guidelines as routines on commercial graphics utilities. Your purchase of a magazine, book, or commercial graphics package allows use of its programs solely for your own, noncommercial use. These routines, therefore, can't be used on disks you're presenting to a client (the exception is a resumé disk used to showcase your work).

Nearly every graphic software producer and every publication has a different usage plan. Some require only a credit line—others require that you, or your client, pay a one-time or an annual licensing fee. If your client is producing software, he or she is already paying Apple an annual fee to distribute DOS 3.3 or ProDOS and may not be willing to incur other expenses.

The decision of which software utilities to use for your graphics is directly tied to the utilities' conditions for usage. Be sure your client agrees with the terms stipulated by the publisher of the software you intend to use for the project.

Where do I go from here?

You'll have to answer this question yourself. We didn't start out as computer professionals nor did we consciously aim for a career with computers—at least not when we purchased our systems. As the power of this incredible tool became evident to us, we jumped in.

Whether computer graphics will be vocation or avocation, be aware that you'll always be learning. You'll constantly run into new terminology and concepts that must be mastered—even if you try to avoid the technical end of things. Unlike more traditional media, computer art and graphics are a moveable feast: new software, new hardware, and new technology.

If you decide to pursue computer graphics as a career, be prepared to spend a great deal of time mastering your craft. The principles of design, composition, and color applied to the more traditional media are just as important in electronic art. The computer will not magically turn you into an artist. To successfully draw and paint with the computer, you must be able to draw and paint *without* a computer.

Programming is no different; you must lay a foundation. Before you can master graphics programming you'll have to study other kinds of programming and dig into those math books. You may be surprised to find that the math that was once so abstract and incomprehensible becomes understandable in the context of graphics—it becomes "real."

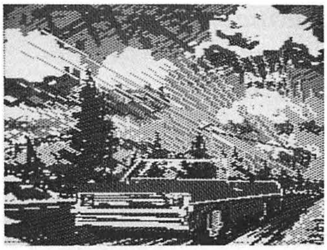
Take courses, read books and magazines, and remember; there's no substitution for hands-on experience.

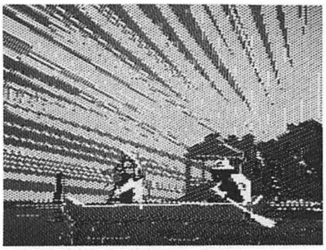
About the artists and their work

From the conception of this book, we knew we wanted to show what artists have done with the Apple computer. Imposing no restrictions, we asked our colleagues to submit work that they felt best represented their specialty or "style". We also requested a short biography and a description of how the works were created.

Here, in alphabetical order, are the artist's bios and comments about their work.

Rush Brown





"Car in a Wooded Landscape," and "Canoe" were done using an Apple IIe with the KoalaPad. The Apple's high resolution capabilities served my purpose well . . .

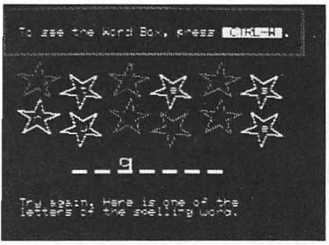
Software provides many useful tools such as adjustable geometric shapes and different marking devices. What makes the process interesting is using these tools in unexpected ways. Working with the computer has forced me to think about how an electronic medium deals with naturalistic subject matter. The result of my musings are electronic, realistic renditions of my inner visions.

I view myself as a painter, draftsman and designer. Twelve years of professional training lead to this self-proclaimed distinction. My formal studies include four years of undergraduate work at Philadelphia College of Art, four years of graduate study at New York University and about three years, collectively, of non-matriculated instruction at California College of Arts and Crafts, the School of Visual Arts, and the New School for Social Research.

I do a lot of work in the sign business and it is here that I was introduced to the use of computer operated machinery for the production of graphics. The "Sign Maker" as it is called, has the capability to produce fonts and universal symbols and cut them out of vinyl or friscate. Recently, Apple developed a program which connects the "Sign Maker" with a graphics tablet, resulting in endless possibilities.

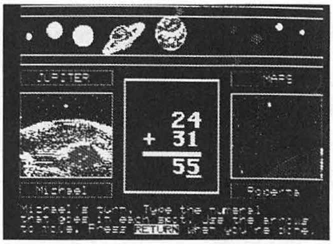
For me, the most exciting characteristic of computer graphics is that unexpected uses for it emerge constantly. My use of computers has proven to be a liberating experience and has altered every aspect of my work.

Michael Callery

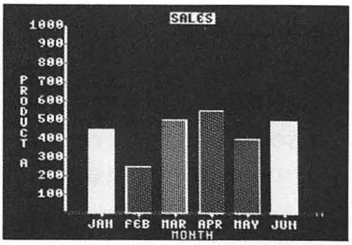


To see the Word Box, press CTRL-W.

Try again. Here is one of the letters of the spelling word.



Michael's turn. Type the numeral that goes in each spot. Use the arrows to move. Press RETURN when you're done.



I was first exposed to computers in the late 60's while finishing an M.S. at Fordham University. In the mid 70's, I was given a sabbatical from my teaching job at Manhattan College to create a pilot computer module for students in my department. Shortly thereafter, microcomputers became widely available. After agonizing over which micro to buy, I chose Apple and haven't looked back since. I resigned my teaching position to work with computers and now teach part-time at The New School for Social Research and Parsons. I've worked with all the major microcomputers but am most skilled on—and enamored with—Apple computers.

My primary interest is education, especially instructional and curricular design. I view computers as central to education and computer graphics as essential to effective educational software. I have programmed or designed educational software projects with many major publishers and worked in the business arena with the consulting firm of McMullen and McMullen.

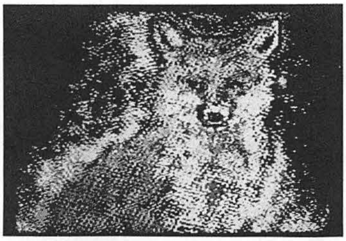
Both of the play screens are from Harcourt Brace Jovanovich curriculum projects and required more information on the screen than I would normally use. To focus on the task, the student input in both games is given in a large, white font and is located near the center of the screen. Instructions are at the bottom of the screen.

"Star Shapes" is an anagram game, students select letters from the stars to spell a word. The screen shown here is seen after the student incorrectly spells the word. A clue letter is shown on the spelling line with the other letters in the star-array. The clue letter then disappears and the student is given another try at spelling the word. The stars are Apple shapes drawn at two rotations. Color is used to highlight the stars that have letters and those that don't. If the student correctly spells a word, a picture of a constellation is shown.

"Planet Hop" is an untimed race game. Two students vie to be the first to reach earth. Students advance to the next planet when they correctly solve an arithmetic problem. The top of the screen shows a diagramatic view of the solar system with small, standard Apple shapes marking the students' location. Color is used to differentiate the players. Windows at each side of the screen show a view of the current planet. These windows are bit-mapped images drawn on the Apple Graphics Tablet and placed on the screen by an assembly language subroutine. When a student correctly answers a question, a "space warp" animation takes place in the student's window. The animation is accomplished by rapidly cycling through five different star scenes in the window.

The first business graph was produced with imaginary data using the default parameters in The Graphics Department. To produce the second graph, I used Fontrix to replace the text with a more attractive font and The Designer's Toolkit to remove the slashed zeros, clarify the axes, and invert the screen.

Ame C. Flynn



Ame C. Flynn has been president of TechniGraphics, a microcomputer graphics firm, for five years. She segued from an extensive education in Fine Arts/Illustration to computer typesetting to computer graphics. She's a member of the faculty at the New School for Social Research where she teaches computer graphics. She has lectured extensively on micrographics, written for several magazines, and produced projects for software houses and television shows.

"Fox" was created in double-high-resolution with an Apple Graphics Tablet, using Dazzle Draw software. I found the hardest thing to depict was the fur, which I did by using both types of spray paint brushes (each gives slightly different effects) and working colors over each other. This allowed me to achieve a fairly rich and furry texture. The ability to mix orange, yellow and grey is a delight.

"Berber" started as a demonstration of the availability of colors in the double-hi-res mode. I again used the graphics tablet and Dazzle Draw. I first filled in the background with grey, so I could work in both black and white. I then tried to work with all sixteen colors. I found it a bit difficult to work on the picture as a whole and wished I could move the menus out of the way while concentrating on my painting. I'm very pleased with both double-hi-res and Dazzle Draw, and I find it exciting to be able to extend the capabilities of my Apple //e.

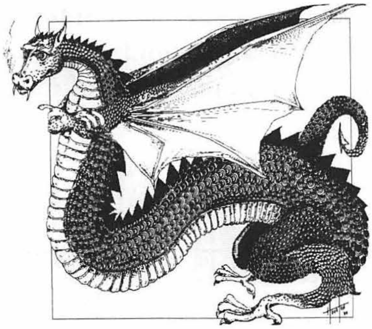
Ken Glickfeld

Ken Glickfeld has been working in many different media over the past 20 years and exhibiting interactive sculpture since 1975. He has had six one-man shows, and exhibited his work in several cooperative shows, invitationals, and museum group shows. He began to work with a microcomputer in 1979 and showed his first computer piece in 1982. He writes about computer subjects and does magazine illustration on his Apple.

I used a variety of programs for the "Left Face" self-portrait series. First I captured several portraits using the Photocaster digitizer. I reworked the images with Designer's Toolkit and then used Beagle Graphics to convert the pictures to double-hi-res. I altered their format and converted the DOS 3.3 images to ProDos in order to rework each image again with Dazzle Draw. Finally, I photographed the images off a composite monitor using Kodachrome film and a 40-R filter.

The computer, as a medium for art making, is still an infant. I feel that we're still trying to determine what unique contribution it can make to the artist's process. I try to be the master of whatever techniques I use and microcomputers permit me to work alone without a staff of technicians. The real-time interactive aspect of a computer makes it possible to create collages that respond to the viewer. The computer permits me to work more freely than I would in other media—I can undo mistakes or repeat an image over and over. I'm able to animate ideas and increase the density of meaning through the added dimension of time.

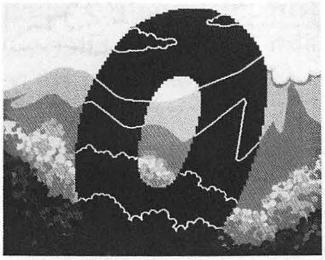
Duke Houston

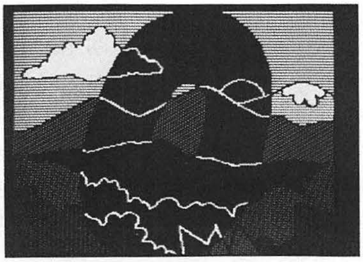


Duke Houston is the Creative Director and Vice President of Development at Data Transforms, a micro-computer graphics software firm, where he first started experimenting with computer generated graphics. With a background in art and cinematography, he was practicing the craft of waiting tables when he stumbled upon the world of microcomputers. He's been happily manipulating mice ever since.

The dragon graphic was created using a mouse and Fontrix software. The image occupies the equivalent of 14 hi-res screens. A rough outline of the dragon was first sketched in with the mouse. The scales were created as font characters and placed within the outline in a manner similar to laying tile or shingling a roof. The final process of shading was accomplished using a cursor "brush" controlled by the mouse, and the gray-scale gradations of various dot patterns available from the software.

Lauretta Jones





Both versions of "And we all fall down! (October)" were created on my Apple IIplus which enjoys the split personality of an added Number Nine graphics board. I began my drawings using the Apple Graphics Tablet for input and the Utopia

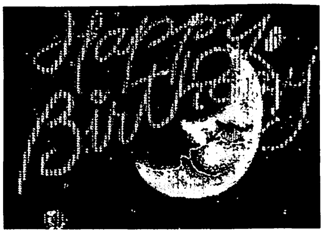
Graphics System to enlarge the letter "O" while I traced it. After cleaning and saving the letter, I finished the Apple version with Designer's Toolkit. Next, I used V-Paint to load the normal Apple screen (the "O") into the Number Nine environment and rework it with more colors and greater resolution. In this case, I emphasized the "jaggies" of the letter to obtain a stronger contrast with the detail of the mountains, clouds, and foliage.

I thought computers fun but irrelevant until by brother interjected a lo-res drawing program into one of our usual sessions of "Raster Blaster" some three years ago. I immediately quit my ad agency job, bought an Apple, and decided I would be a computer illustrator. (It was actually a bit more complicated than that, but I've promised to keep it short.)

My work has appeared as magazine, book, and annual report covers; in ads, packaging, and as a calendar. I've designed logos and created animated software illustrations for education and business. I teach microcomputer graphics at the School of Visual Art in New York and am active in various computer graphic organizations. Recently, I've been combining computer drawings with fabric and stitching.

Computer graphics has increased my vocabulary and my circle of friends. Unfortunately, I am usually so busy with one of my computer projects that I don't get to see very much of them.

M. Brooks Jones



M. Brooks Jones is a promotion graphics designer working primarily in the publishing industry. He has his B.A. degree from Florida State University in Sculpture, and his M.F.A. in conceptual art from Pratt Institute. He has worked with the Apple II computer since August of 1982. He has executed software illustration for several commercially available disks, and is currently developing his own series of floppy disk art pieces. Brooks has studied copyright and copyright issues for computer artists and he lectures and teaches on these topics.

Brooks lives in New York City with his three Apple computers and eight goldfish. He serves as vice-president of the Graphic Artists Guild of New York where he is also chair of the Computer Arts discipline. He is active in the New York chapter of SIGGRAPH.

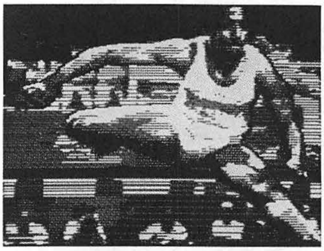
I became enraptured with Apple equipment when I realized that no matter what its limitations—color, resolution, memory, and the inability to do sound and image simultaneously—it gave me the power to fulfill my image fantasies with total control over them from start to finish.

The equipment meant that no matter how primitive the technology that produced my pieces, they would be mine. I've never wanted to work with large systems that involved working with many people to complete a piece. I think one of the things that distinguishes a certain type of artist is the inability to accept the division of labor and with it the outside influence and control it imposes on the work.

I work primarily with two software programs, The Graphics Magician and The Designer's Tool Kit. The Tool Kit is, for me, the most professional paint box to work with; while the Magician gives me animation capabilities. The art that I'm doing now combines work from both programs, along with text, font and sound routines from utilities like Apple Mechanic and Alpha Plot

. . . The computer medium is such a totally visual one that artists will be the contributors who bring it to its fulfillment . . . I believe that the Apple II computer is the first tool that gives all artists the opportunity to become "cult figures" if they choose. I am very excited by that prospect.

Howard L. Kessler

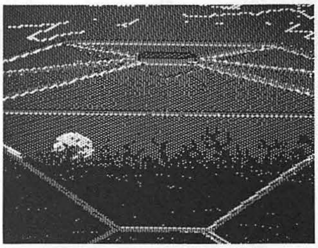


"The Hurdler" is one of a series of computer aided graphics called "Track and Field." The idea of graphic arts being "aided" is not unusual. Few modern graphics begin on the same page on which they are finished. Generally, a collage of figures and text are combined to create a total end product—"The Hurdler" was similarly produced. The male figure was traced from a photo. Then, the tracing was digitized to the size needed, saved, and touched up to its present look. The background, part of a racetrack crowd, was digitized in a simpler form, effecting its distant feeling, and saved so that the figure could be overlayed, cleaned up, and added to, in order to achieve the final realization. Here, the computer was used to photograph at a specific intensity and an electronic mechanical was produced from a number of images and touched up to have a specific style within a specific medium.

I am president of Flourish Design. I've done computer and mechanical graphic art for a number of insurance companies, financial houses, and book companies as well as interior design for commercial and private spaces. I taught computer graphics at the New School/Parsons. I received a Masters degree in Theatrical Design from New York University.

The varied "business of art" has been made simpler, neater, and takes up a much less space with the aid of the computer. Computer graphics has become my dominant direction, and though it is surely not the only art medium I use, it has become my most creative.

Lorene Lavora

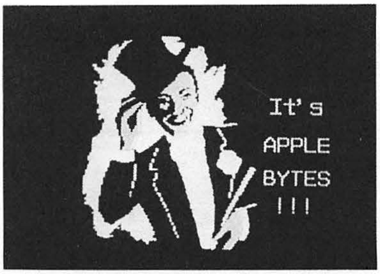


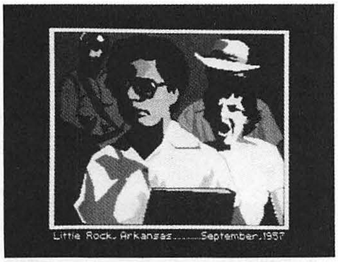
This piece is one of a series designed as a storyboard. The graphics were printed in a small format, using software which allowed their size and color to be altered. They were then pasted up to create a traditional presentation with a unique "techy" look. These same screens were later treated, using simple Apple animation techniques, and videotaped (along with a soundtrack), serving as another mode of presentation. The drawing, "Domed Sunset", was done with an Apple Graphics Tablet using Edu-Paint by Steve Dompier.

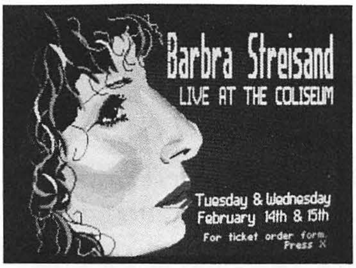
There were several immediate circumstances that seemed to point me in this direction. A formal background in fine art and music, and a love of science were trying to find a place to co-exist at a time that found me in the company of many professional "computer types." They were very encouraging (even though baffling, sometimes). At about the same time, I got a look at a "digital planetarium." This suggested possibilities to me that seemed endless, and so one computer science course led to another. That was around 1980 and since then I've written software and created artwork for children on my Apple //e. I teach computer graphics at the New School for Social Research.

I am currently writing software for a video special effects device called "The Mirage". It's capable of wrapping a screen full of live video images into the 3-D shapes determined by me and the other programmers. Another ongoing project hails back to that digital planetarium. I am designing storyboards, visual effects, and staging for a futuristic musical, "Eutro." It's my hope that this is where art and science will come together to create the stuff of fun and wonder (all interactively, of course).

Maria Manhattan







In January of 1981, I was let loose on a Norpak IPS11 at the Alternate Media Center at NYU. It was my first experience with a computer graphics machine. In spite of thinking that I would not take to this medium because my background was not at all technical, I got hooked almost immediately and haven't stopped working in the field since.

My first piece was a computer comic strip—a fourteen frame cartoon that took advantage of the animation capablities. It was called, "Nancy Reagan Takes the Subway". Since then I have worked with AT&T, NBC, and a variety of others who are experimenting with computer graphics and the videotex field.

Previous to getting involved with this area, I did Ceramic Sculpture for many years and also created a multi-media conceptual performance piece called

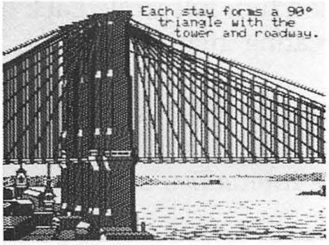
Maria Manhattan's "The Box Lunch" which enjoyed successful runs in both San Francisco and New York City.

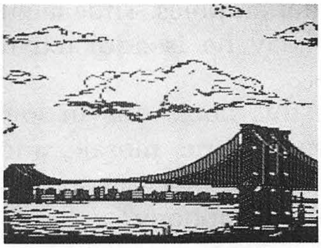
"Fred Astaire", was done for the Apple Bytes project of the Alternate Media Center at N.Y.U. Apple Bytes is a teletext format show broadcast daily on Manhattan Cable TV. The picture was drawn on the Apple II.

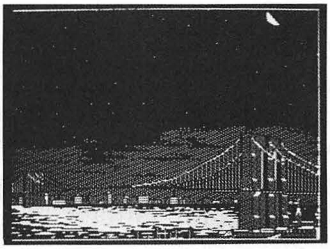
"Little Rock, Arkansas . . . September, 1957", illustrates the integration of Central High School. The picture was taken from a popular news photo of the time. It is not simply a digitized photo however. It was drawn on the Frame Creation System. I used a palette that consisted of many shades of gray, and created the figures by breaking everything down to polygon shapes. I wanted to show how the Videotex medium does not only have to rely on text to convey an idea or emotion. This frame is from a series that I am currently working on called "Pictures from American History".

"Barbra", was drawn on the FCS to illustrate the concept of advertising and being able to order tickets through a videotex terminal.

Jim Markowich



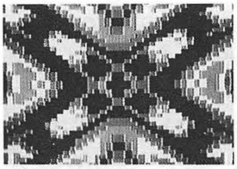


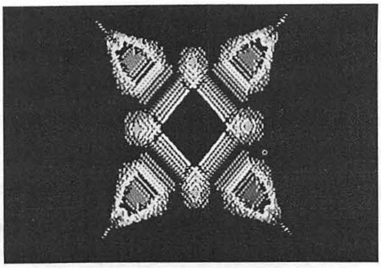


Jim Markowich lives, works, and plays in New York City. He paints, programs, and designs obscure board games.

The graphics from "The Building of the Brooklyn Bridge", were made using the Apple Graphics Tablet and Penguin Software's Complete Graphics System. They were designed to be displayed in monochrome.

Gideon Nettler



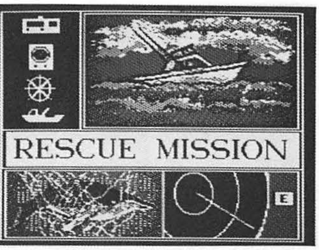


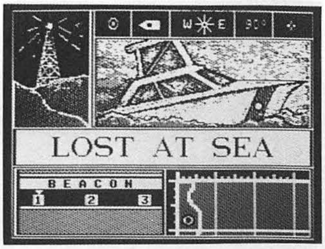
Professor Gideon Nettler, of the Department of Mathematics and Computer Science at Montclair State College, received a BS in Mathematics from Rensselaer Polytechnic Institute and an MS in Mathematics from Polytechnic of New York. He is a published research mathematician, an active member of the New York Academy of Sciences, and chairman of the Nobel Laureate Lectures at Montclair State College where he teaches computer art, robotics, and high technology.

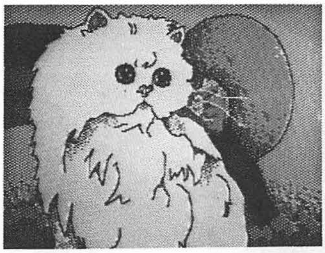
From his early childhood, Professor Nettler had a dream to become an artist. However, practicality drove him first to mathematics, then to computer science, and finally to computer graphics. Thus fate has led him full circle back to his childhood dreams. Today he is addicted to creating mathematically generated computer graphics.

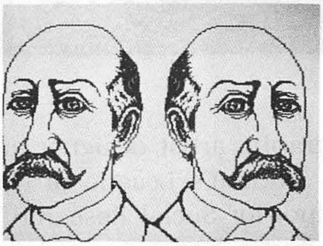
The graphics shown in this book, one in lo-res and one in hi-res, are examples of the endless carpet, fabric, mosaic, and tile shapes and patterns which are mathematically generated by Professor Nettler's interactive computer art program entitled "Symmetry Symphony".

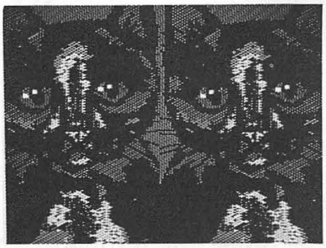
Roberta Schwartz











My niece challenged me to some games of Pacman on her video game system and, although I lost every game, I was fascinated by the graphics. Some other games and a programming cassette further whet my appetite. Advertisements for computers were beginning to appear on television and it wasn't long before I took a sabbatical from teaching art, called a colleague who worked with computers and asked his advice about buying a system. On his recommendation I purchased, sight unseen, an Apple II computer, disk drives, a printer and some graphics software.

Comforted by the salesman's assurance that I couldn't harm the computer by hitting the wrong keys, I indulged myself in a summer of intensive self-instruction, followed by computer graphics courses at The New School for Social Research—I didn't return to my teaching position. I'm delighted with my new career as a freelance computer graphics artist specializing in illustration and animation for educational software and business presentations. I also teach computer graphics, part-time, at the New School and I'm a consulting editor for A+ magazine.

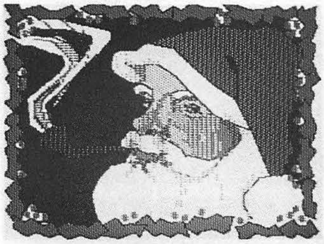
"Rescue Mission" and "Lost At Sea" are two of a series of title, score and win screens created for the Bank Street College of Education's video and software project, "Voyage of the Mimi." Each title screen is a collage of that game's icons and images.

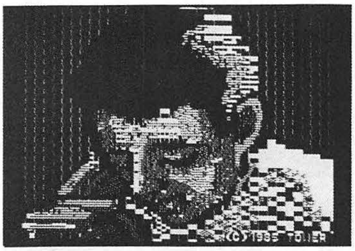
"Over the Rainbow" is a double-resolution sequential picture created with Penguin software's Graphics Magician. Until recently, high-resolution rainbows on the Apple computer were blue, violet, orange and green.

"Night Sky" is a collage of separate images created for a program that allowed the user to press keys to see the constellations. Using Designer's Toolkit and Apple's standard high-resolution, the challenge here was in getting the distinguishing details and shading in the small figures.

"About Face" was drawn with the Gibson light pen, and "Twins" was done with the Apple Graphics Tablet using The Designer's Toolkit software.

Catherine Tower





Catherine Tower, computer graphic artist, designer, and programmer, received her fine art training at the Art Student's League of New York and her computer certification at the New School for Social Research where she is an instructor. Graphics for educational software and animated micro greeting cards are her market trend; business expansion seems inevitable.

"Santa Claus" is a Sequential picture from my animated versions of *'Twas the Night Before Christmas* created for MikRo Greetings. It was done like this:

An Apple IIe with a joystick attached
Gave the screen in hires images unmatched.
Yes, what to my wondering eyes should appear,
A portrait of Santa Claus in colors so dear.
Software by Penguin called The Graphics Magician
In the picture/object editor's position.
More rapid than p-code the sequences replayed
And the commands are keyed in plus the palette's displayed.
Now, line draw! Now, plot brush! Now, zero in and fill it!
On to text! On, delete! On, redraw and save it!
To the top of the poke! To the top of the call!
Now edit it! Edit it! Edit it all!

So with patience and humor anything's possible on the computer.

Appendices and Index

Appendices and Index

ASCII Codes

Codes 0-63 display as flashing, 64-127 as inverse, and 128-255 as normal. PRINTing or keying a control character will execute the ASCII function. To display the character, POKE it onto the screen.

ASCII code	ASCII char	keypress	display+ standard	alternate
0	nul	CTRL-@	@	@
1	soh	CTRL-A	A	A
2	stx	CTRL-B	B	B
3	etx	CTRL-C	C	C
4	eot	CTRL-D	D	D
5	enq	CTRL-E	E	E
6	ack	CTRL-F	F	F
7	bel	CTRL-G	G	G
8	bs	CTRL-H or ←	H	H
9	ht	CTRL-I or TAB	I	I
10	lf	CTRL-J or ↓	J	J
11	vt	CTRL-K or ↑	K	K
12	ff	CTRL-L	L	L
13	cr	CTRL-M or RETURN	M	M
14	so	CTRL-N	N	N
15	si	CTRL-O	O	O
16	dle	CTRL-P	P	P
17	dc1	CTRL-Q	Q	Q
18	dc2	CTRL-R	R	R
19	dc3	CTRL-S	S	S
20	dc4	CTRL-T	T	T
21	nak	CTRL-U or →	U	U
22	syn	CTRL-V	V	V
23	etb	CTRL-W	W	W
24	can	CTRL-X	X	X
25	em	CTRL-Y	Y	Y
26	sub	CTRL-Z	Z	Z
27	esc	CTRL-[or ESC	[	[
28	fs	CTRL-\	\	\
29	gs	CTRL-]	]	]
30	rs	CTRL- ^	^	^
31	us	CTRL-_	_	_

+Not all characters are available on Apple II and IIplus computers with original character generator ROM. Alternate set available only on Apple IIe and IIc computers.

ASCII code	ASCII char	keypress	display standard	alternate
32		▭		
33	!	!	!	!
34	"	"	"	"
35	#	#	#	#
36	$	$	$	$
37	%	%	%	%
38	&	&	&	&
39	'	'	'	'
40	(	(	(	(
41	)	)	)	)
42	*	*	*	*
43	+	+	+	+
44	,	,	,	,
45	-	-	-	-
46	.	.	.	.
47	/	/	/	/
48	0	0	0	0
49	1	1	1	1
50	2	2	2	2
51	3	3	3	3
52	4	4	4	4
53	5	5	5	5
54	6	6	6	6
55	7	7	7	7
56	8	8	8	8
57	9	9	9	9
58	:	:	:	:
59	;	;	;	;
60	<	<	<	<
61	=	=	=	=
62	>	>	>	>
63	?	?	?	?

ASCII code	ASCII char	keypress	display standard	alternate†
64	@	@	@	@
65	A	A	A	A
66	B	B	B	B
67	C	C	C	C
68	D	D	D	D
69	E	E	E	E
70	F	F	F	F
71	G	G	G	G
72	H	H	H	H
73	I	I	I	I
74	J	J	J	J
75	K	K	K	K
76	L	L	L	L
77	M	M	M	M
78	N	N	N	N
79	O	O	O	O
80	P	P	P	P
81	Q	Q	Q	Q
82	R	R	R	R
83	S	S	S	S
84	T	T	T	T
85	U	U	U	U
86	V	V	V	V
87	W	W	W	W
88	X	X	X	X
89	Y	Y	Y	Y
90	Z	Z	Z	Z
91	[	[	[	[
92	\	\	\	\
93	]	]	]	]
94	^	^	^	^
95	_	_	_	_

†On Apple IIc and new model Apple IIe computers the alternate set for ASCII 63-95 displays as MouseText.

ASCII code	ASCII char	keypress	display† standard	alternate
96	`	`		`
97	a	a	!	a
98	b	b	"	b
99	c	c	#	c
100	d	d	$	d
101	e	e	%	e
102	f	f	&	f
103	g	g	'	g
104	h	h	(	h
105	i	i	)	i
106	j	j	*	j
107	k	k	+	k
108	l	l	,	l
109	m	m	-	m
110	n	n	.	n
111	o	o	/	o
112	p	p	0	p
113	q	q	1	q
114	r	r	2	r
115	s	s	3	s
116	t	t	4	t
117	u	u	5	u
118	v	v	6	v
119	w	w	7	w
120	x	x	8	x
121	y	y	9	y
122	z	z	:	z
123	[	{	;	{
124	\	|	<	|
125	]	†	=	†
126	^	~	>	~
127	del	DELETE	?	

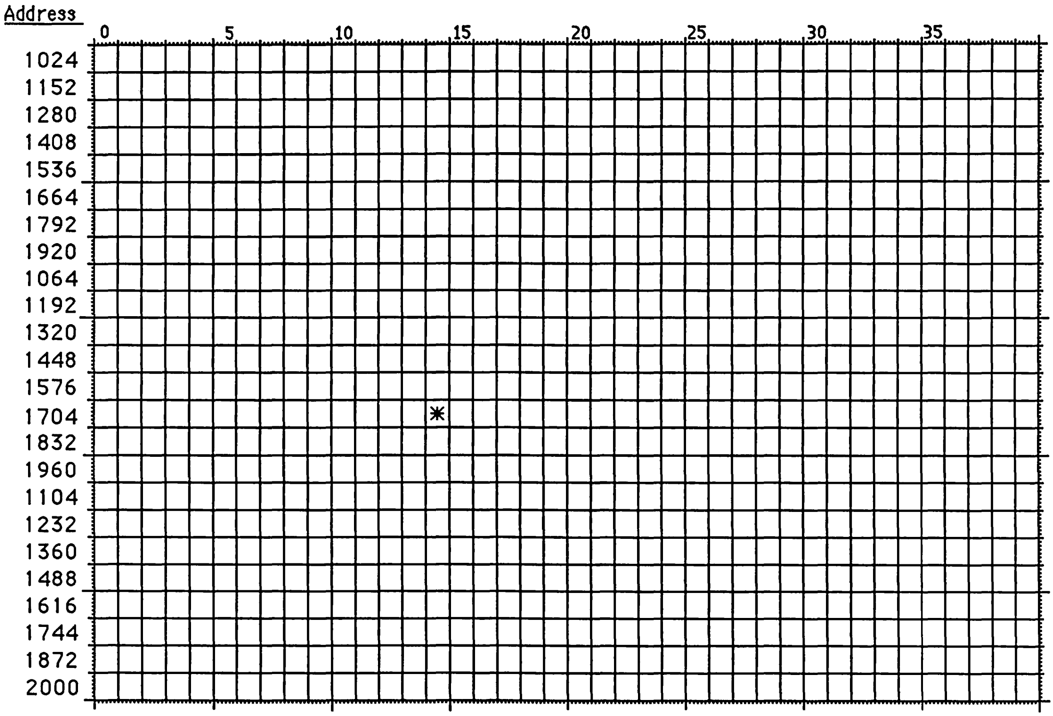
†Primary set displays as inverse; alternate set displays as flashing.

Codes 128-255 follow the standard ASCII sequence and display as normal characters.

Error Codes

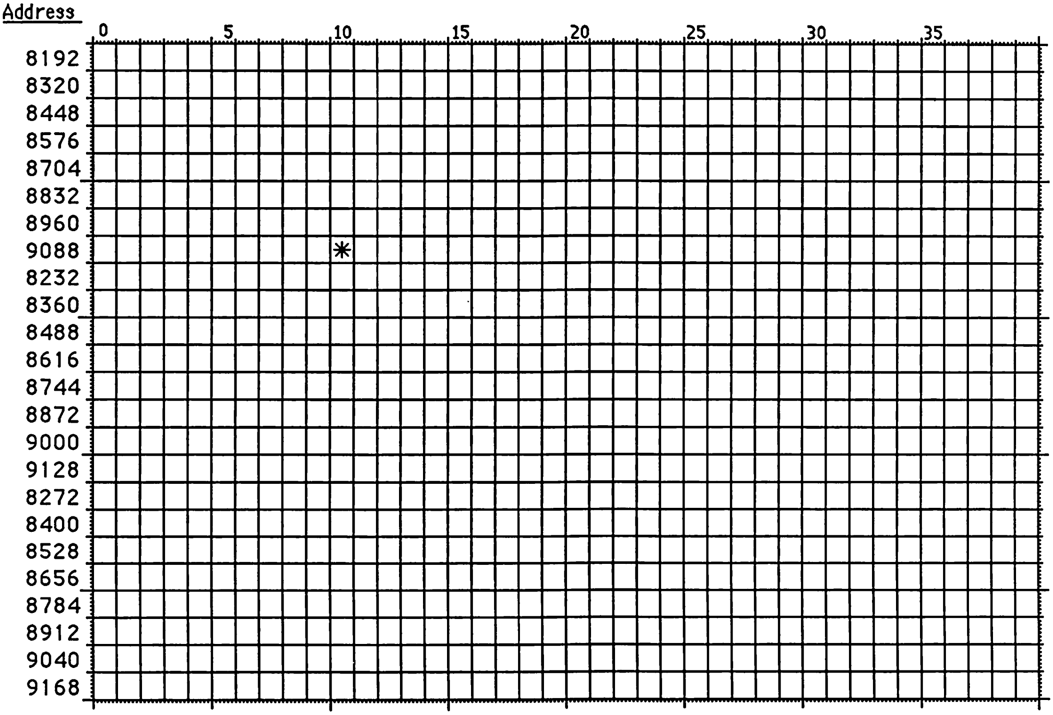
0	?NEXT without FOR	Applesoft
1	Language Not Available	DOS 3.3
2	Range Error	DOS
3	Range Error	DOS
3	No Device Connected	ProDOS
4	Write-Protected	DOS
5	End of Data	DOS
6	File Not Found	DOS 3.3
6	Path Not Found	ProDOS
7	Volume Mismatch	DOS 3.3
8	I/O Error	DOS
9	Disk Full	DOS
10	File Locked	DOS
11	Syntax Error	DOS 3.3
11	Invalid Option	ProDOS
12	No Buffers Available	DOS
13	File Type Mismatch	DOS
14	Program Too Large	DOS
15	Not Direct Command	DOS
16	?Syntax Error	Applesoft
17	Directory Full	ProDOS
18	File Not Open	ProDOS
19	Duplicate File Name	ProDOS
20	File Busy	ProDOS
21	File(s) Still Open	ProDOS
22	?RETURN Without GOSUB	Applesoft
42	?Out Of DATA	Applesoft
53	?Illegal Quantity	Applesoft
69	?Overflow	Applesoft
77	?Out Of Memory	Applesoft
90	?Undefined Statement	Applesoft
107	?Bad Subscript	Applesoft
120	?Redimensioned Array	Applesoft
133	?Division By Zero	Applesoft
163	?Type Mismatch	Applesoft
176	?String Too Long	Applesoft
191	?Formula Too Complex	Applesoft
224	?Undefined Function	Applesoft
254	?Re-enter from start	Applesoft
255	Control-C interrupt	Applesoft
-	Can't Continue	Applesoft
-	Illegal Direct	Applesoft

Lo-res/text page screen map



* Address 1718 (1704+14)

Hi-res Page Screen Map



Within each box:

base address+ 0
+ 1024
+ 2048
+ 3072
+4096
+5120
+6144
+7168

* Base address=9098 (9088+10)
pixel address=base address + vertical offset

PEEK, POKE, and CALLs

All negative addresses may be converted to positive by subtracting them from 65536. e.g. 65536-16304 = 49232.

Display Switches

POKE-16304,0	Display graphics
POKE-16303,0	Display text
POKE-16302,0	Display full-screen
POKE-16301,0	Display split-screen
POKE-16300,0	Display page 1
POKE-16299,0	Display page 2
POKE-16298,0	Display lo-res
POKE-16297,0	Display hi-res
POKE-16290,0	Display double-hi-res (128K IIe & IIc)
POKE-16291,0	Display normal hi-res (128K IIe & IIc)

Text

POKE 32,n	Left edge	0-39 (79)
POKE 33,n	Width	1-40 (80)
POKE 34,n	Top	2-22
POKE 35,n	Bottom	1-24
POKE 36,n	HTAB cursor (40 column)	0-39
POKE 1403,N	HTAB cursor (80 column)	0-79
POKE 37,n	VTAB cursor	0-23
POKE 50,n	Character format: 255=normal, 63=inverse, 127=flash	
POKE 49166,0	Use primary character set (IIc/IIe)	
POKE 49167,0	Use alternate character set (IIc/IIe)	
PEEK(36)	Horizontal cursor position	
PEEK(37)	Vertical cursor position	
CALL-380	Set inverse	
CALL-381	Set flashing	
CALL-868	Clear from cursor to right edge	
CALL-936	Clear whole screen; home cursor	
CALL-958	Clear from cursor to end of screen	

Lo-Res Graphics

PEEK(44)	Line endpoint
PEEK(48)/17	Color code
CALL -1216	Set GR
CALL -1998	Clear lo-res page 1 to black
CALL -1994	Clear top of lo-res page 1 to black

Hi-Res Graphics

PEEK(225)+PEEK(226)*256	X Coordinate-last HPLOT
PEEK(226)	Y Coordinate-last HPLOT
PEEK(228)	Current HCOLOR= setting (0=0, 1=42, 2=85, 3=127, 4=128,5=170,6=213,7=255)
PEEK(231)	Current SCALE= setting
PEEK(232)+PEEK(233)*256	Shape table address
PEEK(249)	Current ROT= setting
CALL -3086	Clear current hi-res page to black
CALL -3082	Clear hi-res to last HPLOTted color
CALL -3106	Set HGR2
CALL -3116	Set HGR

BASIC/DOS

POKE 216,0	Cancel ONERR
POKE -16368,0	Clear keyboard strobe
PEEK(103)+PEEK(104)*256	FP BASIC program start
PEEK(105)+PEEK(106)*256	Beginning of FP variables
PEEK(115)+PEEK(116)*256	Current HIMEM setting
PEEK(175)+PEEK(176)*256	End of FP BASIC program
PEEK(202)+PEEK(203)*256	Int BASIC program start
PEEK(216)	ONERR Active if > 127
PEEK(218)+PEEK(219)*256	Line number with error
PEEK(222)	ERROR code
PEEK(-16384)	Last keypress
PEEK(-16287)	PDL Button 0
PEEK(-16286)	PDL Button 1
PEEK(-16285)	PDL Button 2
PEEK(-16336)	Click Speaker
PEEK(-1101)	Primary Machine I.D. 56 = Apple II, 234 = Apple IIplus, 6 = Apple IIe or IIc
PEEK(-1088)	Secondary Machine I.D. 0 = Apple IIc 56 = Apple IIe 224 = enhanced Apple IIe
CALL 976	Reenter DOS
CALL 1002	Reconnect DOS
CALL -151	Enter System Monitor

Graphics Software and Publishers

Shape Creation

Accu-Shapes	Accent Software
Apple Mechanic	Beagle Bros, Inc.
Complete Graphics System	Penguin Software
Pixit	Baudville
The Artist	Sierra On-Line

Drawing and/or Paint Programs

A.G.I.L. Paint Program	Animation Graphics
Alpha Plot	Beagle Bros, Inc.
Apple Graphics Tablet Software	Apple Computer Inc.
Blazing Paddles	Baudville
Complete Graphics System	Penguin Software
EZ Draw	Sirius Software
Flying Colors with Printout	The Computer Colorworks
MicroIllustrator	Koala Technologies Corp
MicroPainter	Datamost
Paintmaster Scene Utility	Avant-Garde Inc.
Poor Man's Graphics Tablet	Rainbow Computing Inc.
Rainbow Graphics	Rainbow Computing Inc.
The Digital Paintbrush System	The Computer Colorworks
The Designers Tool Kit	Apple/Eclectic Electric

Hi-Res Text Generators

Apple Mechanic	Beagle Bros, Inc.
Auto Graphics	Southwest Ed Psych Services
Flex Type (Flex Text)	Beagle Bros, Inc.
Fontrix	Data Transforms
Higher Text Plus	Apple PugetSound Program Library Exchange
Micro/Typographer	Tidbit Software
The DOS Tool Kit	Apple Computer Inc.

Animation

Movie Maker	Interactive Picture Systems, Inc.
Super Shape Draw and Animate	Avant-Garde Inc.
Take 1	Baudville
The Graphics Magician	Penguin Software
The Graphic Solution	Accent Software

Double Resolution

Beagle Graphics	Beagle Bros, Inc.
Complete Graphics System	Penguin Software
Dazzle Draw	Broderbund
Doublestuff Plus	Doublestuff Software
The Picture Painter	Penguin Software

Light Pen

GIbson Light Pen System	Koala Technologies
Magellan Light Pen System	Magellan Computer Inc.

Graphics Dumps

Image Printer II	Sensible Software
Paper Graphics	Penguin Software
Printographer	Roger Wagner Publishing Inc.
Triple-Dump	Beagle Bros, Inc.
Zoom Grafix	Pheonix Software, Inc.
Color Printer	Enhanced Software Products

Slide Show Utilities

AGIL Super Slide Show	Animation Graphics
Apple Flasher	Crow Ridge Assoc. Inc.
Frame-Up	Beagle Bros, Inc.
Screen Director	BPI
Transitions	Penguin Software

Presentation Graphics

Apple II Business Graphics	Apple Computer, Inc.
Charts Unlimited	Business Information Systems
Graphics Wizard III	Micro Lab Computer Products
Multigraph	Micro Lab Computer Products
The Graphics Department	Sensible Software

Programmer's Utilites

Beagle Graphics	Beagle Bros, Inc.
Cat Graphics	Penguin Software
Doublestuff	Doublestuff Software
Grafmagic	Apple PugetSound Library Program
Graphic Master	Tidbit Software
HGR6	ALF Products Inc.

Graphics Languages

Apple Logo	Apple Computer Inc.
Apple Logo II (128K IIe or IIc)	Apple Computer Inc.
Ceemac	Vagabondo Enterprises
GraForth	Insoft
Terrapin Logo	Terrapin Inc.

CAD/CAM

Cadapple	T & W Systems, Inc.
Build Your Dream House: Architectural Design	Avant-Garde Publishing Corp.
Build Your Dream House: Interior Design	Avant-Garde Publishing Corp.
Hi-Res Electronic Design	Avant-Garde Publishing Corp.
MATC CAD	Milwaukee Area Technical College

Three-dimensional Graphics

The Complete Graphics System	Penguin Software
Apple World Enhanced	United Software of America
Sublogic A2-3D2	SubLogic Communication Corp.

Apple Computer Inc.
20525 Mariani Avenue
Cupertino, CA 95014

Accent Software
4546 El Camino Real Suite S
Los Altos, CA 94022

ALF Products Inc.
1315 F Nelson Street
Denver, CO 80215

Animation Graphics
11317 Sunset Hills Road
Reston, VA 22090

Apple PugetSound Program
Library Exchange
304 Main Avenue S. Suite 300
Renton, WA 98055

Avant-Garde Publishing Corp.
P.O. Box 30160
Eugene, OR 97403

Baudville
1001 Medical Park Dr. S.E.
Grand Rapids, MI 49506

Beagle Bros, Inc.
3990 Old Town Avenue, Suite
102C
San Diego, CA 92110

BPI (Business and
Professional Software, Inc.)
143 Binney Street
Cambridge, MA 02142

Broderbund
17 Paul Drive
San Rafael, CA 94903

Business Information
Systems
5084 Mosiman Road
Middletown, OH 45042

Crow Ridge Assoc. Inc.
P.O.B. 1
New Scotland, NY 12127

Datamost Inc.
8943 Fullbright Avenue
Chatsworth, CA 91311

Data Transforms
616 Washington Street
Denver, CO 80203

Doublestuff Software Inc.
2053 West 11 Street
Brooklyn, NY 11223

Enhanced Software Products
P.O. Box 178
Wantagh, NY 11793

Insoft
10175 Southwest Barbur Blvd.
Suite 202B
Portland, Oregon 97219

Interactive Picture Systems,
Inc.
270 Park Avenue South #6A
New York, NY 10010

Koala Technologies Corp
3100 Patrick Henry Drive
Santa Clara, CA 95050

Magellan Computer Inc.
4371 E. 82 Street #D
Indiana, IN 46250

Micro Lab Computer
Products
2699 Skokie Valley Road
Highland Park, IL 60035

Milwaukee Area Technical
College
1015 North 6th Street
Milwaukee, WI 53203

Penguin Software
830 4th Avenue
P.O. Box 311
Geneva, IL 60134

Pheonix Software, Inc.
64 Lake Zurich Drive
Lake Zurich, IL 60047

Rainbow Computing Inc.
8811 Amigo Avenue
Northridge, CA 91324

Roger Wagner Publishing Inc.
10761 Woodside Avenue,
Suite E
Santee, CA 92071

Sensible Software
210 S.Woodward, Suite 229
Birmingham, MI 48011

Sierra On-Line
Sierra On-Line Bldg
Coarsegold, CA 93614

Sirius Software
10384 Rockingham Drive
Sacramento, CA

Southwest Ed Psych Services
P.O.B. 2231
Goleta, CA 93118

SubLogic Communication
Corp.
713 Edgebrook Dr.
Champaign, IL 61820

T & W Systems, Inc.
7372 Prince Drive
Huntington Beach, CA 92647

Terrapin Inc.
222 3 Street
Cambridge MA 02142

The Computer Colorworks
3030 Bridgeway
Sausalito, CA 94965

Tidbit Software
P.O. Box 5579
Santa Barbara, CA 93108

United Software of America
750 Third Avenue
New York, NY 10022

Vagabondo Enterprises
1300 East Algonquin
Schaumburg, IL 60195

Resources

Organizations

Association for Computing Machinery (ACM) / SIGGRAPH. Membership inquiries: ACM, 11 West 42nd St., New York, NY 10036. (212) 869-7440.

National Computer Graphics Association (NCGA) Inquiries: NCGA, Suite 601, 8401 Arlington Blvd., Fairfax, VA 22031. (703) 698-9600.

Your local Apple User Group may also have a special interest group (SIG) for graphics. We highly recommend connecting with your local user group.

Magazines: Graphics

Computer Graphics World 1714 Stockton St., San Francisco, CA 94133. (415) 398-7151.

Computer Pictures Backstage Publications, 330 West 42nd. St., New York, NY 10036.

Magazines: Apple-Specific

A+ Subscription Inquiries: A+, PO Box 2965, Boulder, CO 80322.

InCider Subscription Inquiries: InCider, PO Box 911, Farmingdale, NY 11737.

Nibble Subscription Inquiries: Nibble, 45 Winthrop St., Concord, MA 01742.

Magazines: General

Popular Computing Subscription Inquiries: Popular Computing, PO 307, Martinsville, NJ 08836.

Byte Subscription Inquiries: Byte, PO 596, Martinsville, NJ 08836.

Bailey, H.J. and J.E. Kerlin. 1984. **Apple Graphics Activities Book.** Brady.

The reader is assumed to know a bit about the Apple and about programming although the programs toward the beginning of the book are simple. The book covers beginning graphics programming but does not include interactive graphics, graphics input or output devices, or any discussion of available utility software.

Dewitt, W.H. 1984. **Art and Graphics on the Apple II/IIe**. Wiley.

This book consists primarily of small, elementary examples, some of them quite good.

Franklin, H.M., J. Koltnow, and L. Finkel. 1982. **Golden Delicious Games for the Apple Computer**. Wiley.

This is one of the best books in the "sample program" category. The authors discuss some concepts, but their primary focus is to explain, by example, how to create simple games incorporating graphics and sound.

Korites, B.J. 1981. **Graphic Software for Microcomputers**. Kern.

This book deals extensively with 3-D concepts: hidden line removal, transformations, rotations, and so on. There is no discussion of color, of shapes, of text, of nearly anything that one would want to do with graphics aside from the 3-D aspects. The programs are written in very simple Applesoft, appropriate for the beginner.

Meyers, R. 1982. **Microcomputer Graphics**. Addison-Wesley.

Like Hearn and Baker, Meyers covers computer graphics concepts, including 3-D, hidden-line removal, and clipping. The examples, written in Applesoft, are of immediate use to the Apple user. The book looks formidable to beginners because the author presents the mathematical backgrounds of his routines.

Person, Ron. 1985. **Animation Magic**. Osborne/McGraw Hill.

This book focuses specifically on shape table animation and includes the source listings (in Applesoft) of several utility programs. These programs cannot subsitute for commercial utilities but have the distinct advantage of being LIST-able and modifiable by the reader. Recommended for those people who are interested in programming.

Stanton, J. 1982. **Apple Graphics & Arcade Game Design**. The Book Co.

We recommend Stanton to advanced programmers interested in the "guts" of Apple graphics. It is the only book that deals exclusively with assembly language and with the techniques needed to create animated games. The book describes the technical aspects of the Apple screen and Apple color very clearly and the routines presented are useful to anyone using assembler for graphics.

Waite, M. 1979. **Computer Graphics Primer**. Sams.

Outdated but the Apple-specific examples are good and are explained well.

Williams, K., R. Kernaghan, and L. Kernaghan. 1983. **Apple II Computer Graphics.** Brady.

This book is a technical book and very specific to the Apple. The authors start at the core of the Apple with a discussion of the Apple System Monitor which may intimidate some readers. The backgrounds of the authors (game programmers/designers) is obvious.

Books: General Graphics

Artwick, B. 1984. **Applied Concepts in Microcomputer Graphics.** Prentice-Hall.

Artwick is one of the creators of the subLOGIC 3-D system. His book is especially good in its discussion of the hardware aspects of comptuter graphics.

Foley, J.D. and A. Van Dam. 1982. **Fundamentals of Interactive Computer Graphics**. Addison-Wesley.

Probably the most widely used technical computer graphics text book. It is comprehensive and, for intermediate or advanced programmers, understandable.

Giloi, Wilfgang K. 1978. **Interactive Computer Graphics: data structures, algorithms, languages**. Prentice-Hall.

The source book for graphic algorithms. Highly technical and highly recommended for the technical bookshelf.

Harrington, Steven. 1983. **Computer Graphics: a Programming Approach**. McGraw-Hill.

Covers the same area as Giloi but geared toward the intermediate programmer.

Hearn, D. and M. P. Baker. 1983. **Microcomputer Graphics**. Prentice-Hall.

Good introduction to the technical aspects of graphics programming intended for the programmer. This is the only book in this section that can be reasonably approached by a beginner.

Newman, William M. and R. F. Sproull. 1973. **Principles of Interactive Computer Graphics**. McGraw-Hill. (Second edition, 1979)

This is the classic.

Books: Graphics Picture Books

Cohen, H., B. Cohen, and P. Nii. 1984. **The First Artifical Intelligence Coloring Book.** Kaufmann.

A unique book drawn entirely by a series of increasingly "intelligent" computer programs. Although the printouts are not traditional computer graphics, they represent an interesting glance at one possible future for the field.

Jankel, A. and R. Morton. 1984. **Creative Computer Graphics**. Cambridge.

A lavishly produced look at the full spectrum of computer graphics. The authors have also included a wide range of graphics concepts in the text, however the real attraction of this book is the abundance of color plates.

Pruett, M. 1984. **Art and the Computer**. McGraw Hill.

Another lavish production but with a less comprehensive scope than Jankel and Morton. Pruett is one of the pioneers in the field; his text reflects his experiences.

Scott, J., ed. 1984. **Computergraphia**. Gulf.

Written by the Third Coast Computer Graphics Group, this book has lots of color plates and covers a wide range of computer graphics applications.

Index

Apple Graphics:

TOOLS AND TECHNIQUES

Michael L. Callery · Roberta Schwartz

Should I use lo-res or hi-res graphics?
How does the Apple produce double hi-res color?
How can I get colored text on the Apple?
How do I do computer animation?
What makes a graphic software package "good"?
What do I draw and paint with?

These are just a few of the provocative questions addressed in **Apple Graphics: Tools and Techniques.** The answers to these questions are part of a practical, useable approach that cuts through computer jargon and technical details. The authors explore the routines, techniques, and materials used to create graphics with an Apple II, IIe, or IIc computer.

You will learn about Apple computers, programming, and fundamental graphics concepts. You will see graphics created by professional Apple computer artists and learn about the commercial software and hardware tools that they use. **Apple Graphics: Tools and Techniques** examines these tools—how they work and what they can do for you. With simple examples, this fully illustrated book will teach you how to use an extraordinarily versatile medium for your creative expression.

With **Apple Graphics: Tools and Techniques,** you'll be well on your way toward producing effective, professional microcomputer graphics with an Apple computer.

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

ISBN 0-13-039512-9